Variable Length Representation in Evolutionary Electronics

Ricardo S. Zebulum[*]
CCNR, Biology School
University of Sussex
Brighton BN1 9QG, UK
ricardoz@cogs.susx.ac.uk

Marco Aurélio Pacheco
ICA, Depto de Eng. Elétrica
Universidade Católica
Rio, 22453-900, Brazil
marco@ele.puc-rio.br

Marley Vellasco
ICA, Depto de Eng. Elétrica
Universidade Católica
Rio, 22453-900, Brazil
marley@ele.puc-rio.br

Abstract

This work investigates the application of variable length representation (VLR) evolutionary algorithms (EAs) in the field of Evolutionary Electronics. We propose a number of VLR methodologies that can cope with the main issues of variable length evolutionary systems. These issues include the search for efficient ways of sampling a genome space with varying dimensionalities, the task of balancing accuracy and parsimony of the solutions, and the manipulation of non-coding segments. We compare the performance of three proposed VLR approaches to sample the genome space: Increasing Length Genotypes, Oscillating Length Genotypes, and Uniformly Distributed Initial Population strategies. The advantages of reusing genetic material to replace non-coding segments are also emphasized in this work. It is shown, through examples in both analog and digital electronics, that the variable length genotype's representation is natural to this particular domain of application. A brief discussion on biological genome evolution is also provided.

Keywords

VLR evolutionary algorithms, evolutionary electronics, genetic memory, digital circuits, analog filters.

1 Introduction

Variable length representation (VLR) aims to give to artificial evolutionary systems the capability of sampling a genome space corresponding to phenotypes of different complexities. There are basically two concepts in nature that have not yet been adequately simulated in artificial evolutionary systems: the impressive ability of complex organisms to form from simple ones (Harvey, 1993) and the existence of both coding and non-coding genetic material (Ridley, 1996). The use of variable length data structures representing the genotypes in evolutionary algorithms (EAs) allows the exploration of these two biological concepts.

In this article, we devise a number of techniques that utilize VLR evolutionary algorithms and apply them to the novel field of Evolutionary Electronics (EvolElec) (Sipper,

[*]The author is currently a post-doctoral scholar at the NASA Jet Propulsion Laboratory.

1997). This new area of study has arisen from the increasing number of EA applications in the electronic engineering domain. Evolutionary Electronics is an attractive testbed for methodologies involving variable length genotypes (VLGs), due to the large number of practical applications in this area. EvolElec can derive benefits from using the VLR concept; particularly, in the domain of applications covered by this work, where evolutionary systems are used to design circuits of unknown sizes. This requirement, together with parsimony, which is a key feature for solutions in engineering design, is well suited to EAs using variable length representation. We examine the following cases: evolution of digital circuits (comparators, multiplexers, and parity detectors) and evolution of analog circuits (passive filters and bipolar transistor amplifiers).

The methodologies proposed in this work cope with the basic issues of variable length genotypes: sweeping the genome space and controlling the size or complexity of the solution. We present three different approaches for sampling genome spaces of varying dimensionalities: Increasing Length Genotypes (ILG) (Harvey, 1993), Oscillating Length Genotypes (OLG), and Uniformly Distributed Initial Population (UDIP). We also examine two classes of mechanisms to control the solution complexity: implicit control mechanisms, where the evolutionary system develops its own regulation of the genotype size; and explicit mechanisms to control the solution size, such as the Pareto Optimal concept (Fonseca and Fleming, 1995). Finally, we identify a way of reusing useful genetic material in the replacement of the non-coding loci of the chromosomes by including a memory data structure in the evolutionary system. As we shall show, this mechanism models the advantageous preservation of genetic material associated with diploid organisms (Ridley, 1996).

This work is divided into five additional sections. Section 2 introduces the field of Evolutionary Electronics and discusses the reasons why VLR evolutionary systems constitute a natural approach to this field. Section 3 presents the basic methodologies developed by the authors to deal with the features of the genome space characteristic of VLR evolutionary algorithms. Section 4 presents the case studies in analog and digital electronics, as well as a discussion of the methodologies used in each case. Section 5 presents a brief discussion on the biological aspects that stimulate us to investigate this new class of EAs. Section 6 concludes the work.

2 Evolutionary Electronics

Evolutionary Electronics (EvolElec) applies the concepts of genetic algorithms to the evolution of electronic circuits (Thompson, 1996). The main idea behind this research field is that each possible electronic circuit can be represented as an individual or a chromosome of an evolutionary process, which performs standard genetic operations over the circuits. Due to the broad scope of the area, researchers have been focusing on different problems, such as placement (Esbensen and Mazumder, 1994), routing (Esbensen, 1994), Field Programmable Gate Array (FPGA) mapping (Kommu and Pomenraz, 1993), optimization of combinational and sequential digital circuits (Miller and Thomson, 1995), synthesis of digital circuits (Miller et al., 1997), synthesis of passive and active analog circuits (Koza et al., 1996), synthesis of operational amplifiers (Kruiskamp and Leenaerts, 1995), and transistor size optimization (Rogenmoser et al., 1996). Of great relevance are the works focusing on "intrinsic" hardware evolution (Layzell, 1998; Thompson et al., 1996) in which fitness evaluation is performed in silicon, allowing a higher degree of exploration of the physical properties of the medium. This particular area is frequently called Evolvable Hardware, though this terminology may also be used in a broader sense.

Let us first consider the problem of sizing[1] the target circuit that is to perform a desired function. If classical genetic algorithms with fixed length genotypes are used, then a previous knowledge will be required to set the size of the target system; for instance, the number of transistors when the target design is an analog amplifier, or the number of boolean gates when the target design is a digital circuit. Besides requiring a previous knowledge from the user, fixed length representation limits the potential of evolutionary algorithms in finding not only the solution itself but also its size. It is, therefore, expected that variable length genotypes will be better suited for Evolutionary Electronics (Thompson et al., 1996).

In our VLG approach, each genotype G is as a collection of genes divided into two subsets: the subset of active genes and the subset of inactive genes. By active or coding genes we mean those genes being considered by the EA when evaluating the chromosomes' fitness. Equation 1 describes our concept of genotype.

$$G = \bigcup {g_i}^{M_i}; \qquad i = 1, ..., n; \quad \textit{and} \quad M_i = 0 \quad \textit{or} \quad M_i = 1 \tag{1}$$

The above equation defines our genotype model as a collection of n genes, and each gene g_i can be classified in either an active gene, for $M_i = 1$, or an inactive gene, for $M_i = 0$. We define the number of active genes by the variable m and the number of inactive genes by the variable l (m and l are formally defined in Section 3.1). Our proposal is to keep the values of m and l variable along the evolutionary process, while keeping the total genotype length, $n = m + l$, constant. We will call the number of active genes m the effective genotype length. Each gene will encode for a piece of the electronic circuit or phenotype; this piece may be a discrete component, such as a resistor, capacitor, or transistor, or a mathematical function performed by a set of electrical components, such as a product of boolean variables.

Our assumption is that we may employ similar VLR methodologies to different problems in EvolElec, ranging from boolean function synthesis to the design of high gain amplifiers, since, in all these problems, the size of the circuit is part of the solution.

3 VLR Methodologies

VLR evolutionary algorithms have been intensively investigated since the beginning of the 90's. Examples of these algorithms are the Messy GAs (Goldberg et al., 1990), LS-1 classifiers (Smith, 1980), genetic programming (Koza, 1992), and the Species Adaptation Genetic Algorithm (SAGA) (Harvey, 1993). A survey of these algorithms may be found in Harvey (1993).

In this section, we describe our VLR methodologies, which borrow some concepts from the SAGA methodology. The following issues are addressed below: genotype model, sweeping strategies, genotype size control, and the use of non-coding segments.

3.1 Genotype Model

Figure 1 illustrates the genotype model employed in this work, which is based on the concept of active and inactive genes described in Equation 1. The model adopts a genotype formed by a string, as in standard genetic algorithms. The string is a collection of genes,

[1] In the context of this work, the word "sizing" is employed to express the number of components of the target circuit.

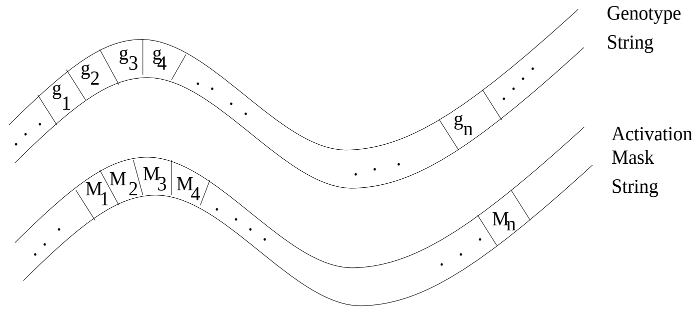

Figure 1: Genotype model adopted throughout this work.

g_1, g_2, ... g_n, each one with the same number of loci L that is usually greater than 1 and dependent on the application. Our model differs from standard genetic algorithms in that it employs another string linked to the main one called the *activation mask string*. The bits of this auxiliary string work as masks for the genotype main string. A particular gene g_i in the main string will be active if the associated value M_i in the mask string equals 1. If M_i equals 0, then the gene g_i will be inactive. This simple mechanism is used throughout this work to provide a computational implementation for our VLR evolutionary algorithm. According to the representation used in Equation 1, the effective genotype length m is given by the number of active genes (M_i = 1):

$$\mathrm{m} = \sum_{i=1}^{n} M_i \quad ; \quad n = l + m \tag{2}$$

In our experiments, we have been careful to use a value of n big enough to avoid upper limit boundary effects (Banzhaf et al., 1998).

Besides the standard evolutionary algorithm operators, selection, crossover, and mutation, we have included another genetic operator called *vary-length* operator, which can either activate or deactivate a particular gene g_i by changing the value of M_i. This operator is applied to each locus of the activation mask string with a particular probability, and it is not particularly inspired in any *explicit* mechanism used in nature; rather, it is a way to provide sampling over the whole genome space consisting of phenotypes of different complexities. The concept of genotype masks has been used in genetic algorithms as a way of preparing for uniform crossover (Goldberg, 1989); here, we employ the same concept within a different context.

In addition to the vary-length operator, the *crossover* operator will also modify the genotypes' masks. When two genotypes exchange genetic material through crossover, the correspondent masks of each gene will be similarly exchanged. We have used the standard one-point crossover operator.

Exponential selection (Bäck, 1994; Goldberg and Deb, 1991) has been used in our experiments, since it provides an easier way to control the selection pressure.

3.2 Sweeping Strategies

Sweeping strategies refer to the way in which the different dimensionalities of the genome space are sampled by the EA. We devised three strategies to handle this problem: Increasing Length Genotypes (ILG), Oscillating Length Genotypes (OLG) [2], and Uniformly Distributed Initial Population (UDIP).

3.2.1 Increasing Length Genotypes (ILG)

This strategy has been devised within the context of the SAGA methodology, and it seems to be the most suitable for engineering problems. The Increasing Length Genotypes strategy is inspired by the biological formation of complex organisms from simple ones (Ridley, 1996). We initialize all the genotypes with the same activation mask in which just a few bits are set to one; hence, just a few genes are initially active, and simple phenotypes are produced initially. Except for the fact that we fix a minimal number of bits set to 1 in the initial mask, its general pattern is randomly initialized. Since the genotypes only increase in size, the vary-length operator may only activate genes in this approach.

There are two basic advantages in using the ILG strategy: the implicit control mechanism for the solution sizes and the gain in speed.

The implicit control mechanism is characterized by the fact that the average population genotype size usually stabilizes in a particular value, instead of growing until the upper bound limit of the genotype. This is due to the fact that, once solutions are found with a particular size, selection will start to check individuals larger than this optimal or sub-optimal size. Additionally, it has been observed in our experiments that the size of the evolved solutions is very sensitive to the vary-length operator expression, which gives the rate at which this operator is applied to each mask bit of the activation string. It will be shown in the case studies that a slowly decaying exponential function is used to determine the rate of application of this operator.

The gain in speed in the ILG strategy stems from the fact that circuits with complexity much beyond the one of the optimal solution, whose simulation is more time consuming, are not sampled.

3.2.2 Oscillating Length Genotypes Strategy (OLG)

The OLG strategy is a variation of the ILG strategy in which the genotypes are also allowed to decrease in size. The vary-length operator now switches the genes, activating and deactivating the mask bits. One possibility is to assign to this operator a rate obeying a sinusoidal function. The main purpose of this strategy is to create pathways from large to smaller genotypes with similar fitness values.

3.2.3 Uniformly Distributed Initial Population Strategy (UDIP)

Instead of starting with a population of small genotypes, the initial genotype sizes are now randomly assigned to values ranging from one to the maximum number of genes n. Therefore, the initial population is heterogeneous in terms of genotype size. The vary-length operator rate is assigned to a constant and low value. The objective of this strategy is to find out the EA behavior with heterogeneous populations.

[2] Suggested by Anil Seth, Centre for Computational Neurosciences and Robotics, University of Sussex.

3.3 Genotype Size Control

Genotype size control relates to the problem of balancing the accuracy and parsimony of the solutions. Zhang (1997) devises a taxonomy of growth control schemes, classifying them into direct and indirect, non-adaptive and adaptive, and off-line and on-line. Within the genetic programming framework, many approaches, such as, adaptive parsimony pressure (Zhang and Mühlenbein, 1995), deleting crossover (Blickle, 1996), and explicitly defined introns (Nordin et al., 1995) have been proposed to tackle the bloating phenomenon (Langdon, 1997). Another more general approach is the Minimum Description Length (MDL) methodology (Iba et al., 1997). As we have previously stated, some sweeping strategies can provide a solution size control mechanism for free. Within our EA framework and concerning our area of application, it has been verified that implicit mechanisms produce good results. However, for the sake of comparison, we propose the use of the Pareto-Optimal concept for multi-objective optimization (Fonseca and Fleming, 1995) as a method to balance parsimony and correctness.

The main advantage of using Pareto is that we do not need to incorporate terms of correctness and parsimony into the fitness evaluation function. Instead, at each generation, a list of individuals that are dominated or non-dominated (Fonseca and Fleming, 1995) by correctness and parsimony is produced, and their fitness values are changed at periodic intervals of generations. This change consists of assigning the lowest possible fitness value to all dominated individuals, regardless of their actual fitness, and maintaining the fitness values of the non-dominated individuals. Depending on the selection method employed, the dominated individuals may be partially or totally eliminated from the reproduction process. The period at which the fitness values are changed affects the selection pressure; reducing or increasing this period will respectively increase or reduce the selection pressure towards fit and parsimonious solutions. After testing different values, we have kept this period at 20 generations.

3.4 Use of Non-coding Segments and the Memory Paradigm

Within the genetic programming framework, non-coding segments have been studied from the bloating phenomenon perspective, i.e., tree size explosion accompanied by a reduced performance of the optimization process (Blickle, 1996). According to Banzhaf et al. (1998), the advantage of variable size genomes is their ability to choose their own representation, whereas the apparent disadvantage is the production of non-coding segments. They concluded that while introns do not affect the fitness of the individual directly, they may affect the likelihood that the individuals will survive by protecting them against destructive crossover. In another work, Wu and Lindsay (1995) performed an empirical investigation of the influence of non-coding segments in the performance of genetic algorithms.

Another implicit role of non-coding segments in artificial evolutionary systems relates to the formation of neutral networks. Harvey and Thompson (1997) investigated this implicit functionality of non-coding segments in the context of Evolutionary Electronics. They have used the concept of neutral networks of fitness landscapes in the evolution of FPGA circuits. A neutral net is a connected pathway through the genotype space, which can be transversed through the application of genetic operators such as mutation while not changing the fitness. These networks are formed if the genetic encoding has a number of potentially useful redundant loci (called "junk DNA") that are functionless within a particular context, but, given different values elsewhere, they may well become significant.

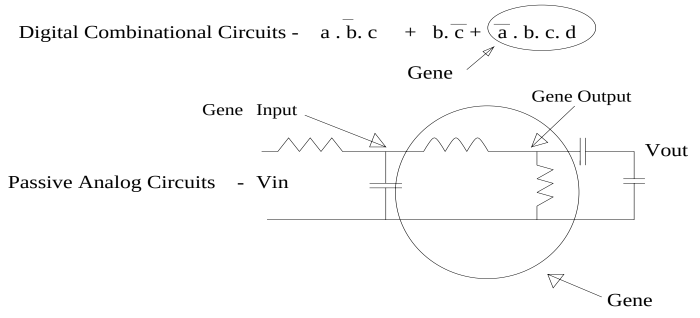
Digital Combinational Circuits - $a \,.\, \bar{b}.\, c \;+\; b.\, \bar{c} + \bar{a} \,.\, b.\, c.\, d$
Gene
Gene Input
Gene Output
Vout
Passive Analog Circuits - Vin
Gene

Figure 2: Chromosome representation and memory paradigm in evolutionary electronics: digital and analog cases.

It has been suggested that neutral networks played an important role in the evolution of a two-tones recognizer circuit (Thompson, 1997).

In our investigation, we make an **explicit** use of non-coding segments; instead of randomly initializing the genes' contents as they are being activated, we pick out a gene known a priori to be probably fitter. In order to use this approach called *memory paradigm*, our representation will have to be such that the evaluation of each gene as an individual unit will have to be enabled (Gero, 1998).

In Evolutionary Electronics, this kind of representation can be achieved in both digital and analog circuits. In the former, each gene may account for a boolean product of input variables and the whole genotype will encode for the overall sum of products representing the circuit. In the latter, each gene may encode for a mesh [3] of the circuit. This is shown in Figure 2.

The memory paradigm requires the existence of a central data structure in the EA, which will behave as a genetic memory; as fitter genes come up in the beginning of evolution, they are stored and then reused as the genotypes grow in size. Evolutionary algorithms using the memory paradigm will have the following advantages over conventional ones:

1. Overcoming schemata processing limitations (Goldberg, 1989) of classical GAs, since the highly fit, short-sized schemata initially spread throughout the population are now *explicitly* gathered to constitute longer sequences of genes;

2. Saving useful genetic material from being lost once it appears in low fitness genotypes, since it is now stored in the memory. This is akin to the diploidy effect of preserving genetic material in natural evolution (Goldberg, 1989).

The main consequence of the use of the memory paradigm and its associated representation is the departure from the implicit schemata processing concept of traditional genetic algorithms. In the case of building blocks larger than the gene size, though, the implicit schemata theorem still holds at the gene level.

Another advantage of this representation is the possibility of carving up the genome

[3]A mesh is defined here as an arrangement of two components. For instance, Figure 2 shows an analog circuit made up of three meshes.

space. To each gene g we may assign a unique index I given by:

$$I(g_l) = \sum_{i=0}^{n-1} L_i \, K^i \qquad (1 \leq l \leq K^n) \tag{3}$$

where L_i is the value of the locus i of a n-loci gene, and K is the representation alphabet cardinality (assuming binary or integer representation). There are K^n different genes. A huge genome space can then be uniformly decomposed into S slices according to the genes' assignments, and each parallel running evolutionary algorithm $EA_j (1 \leq j \leq S)$ will sample a region R_j that includes a particular range of indexes:

$$\frac{K^n}{S}(j-1) \leq R_j \leq \frac{K^n}{S} j \tag{4}$$

A simple penalty term is applied in each EA_j fitness evaluation function, to induce them to sample their respective slices of the genome space. This penalty holds for all genes in a particular chromosome so that the overall genotype will tend to stay at its EA correspondent slice. The final form of the fitness equation will be presented in Subsection 4.1.

4 Case Studies

This section presents the results of the application of the above concepts in two major areas of electronics: digital and analog.

4.1 Digital Circuits

We examine here the application of VLR EAs in the synthesis of combinational digital circuits. Given a truth table [4] or a list of minterms [5] for a particular boolean function, the task is to find a minimum digital circuit that implements this function. The evolution of combinational functions has been classically used as a testbed for evolutionary methodologies (Koza, 1994). Evolutionary algorithms may be used in combinational function optimization together with simplification heuristics, providing results that are competitive with the state-of-the-art in CAD (Computer Aided Design) (Miller and Thomson, 1995). Here we handle the problem of boolean function synthesis without any heuristics, which turns out to be a hard and interesting problem (Miller et al., 1997).

4.1.1 Problem Modeling

Representation

Considering that any boolean function can be represented as a sum of product terms, we have represented the genotypes as a collection of genes that encode for any possible product of boolean variables, complete or incomplete. For a $n - input$ digital circuit, each gene will have n loci that can take three possible values: 0 if the correspondent input variable is complemented, 1 if the input variable is uncomplemented, and 2 if the input variable is absent. This can be seen in Figure 3, showing the representation of a

[4] List containing 2^n output values for a n-input boolean function.

[5] Product of boolean terms containing as many literals as there are variables in the boolean function (Hill and Peterson, 1981)

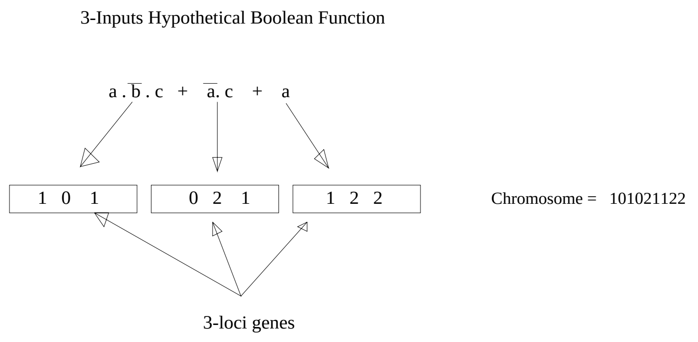

Figure 3: Genotype mapping for a hypothetical boolean function.

hypothetical boolean function consisting of three product terms resulting in a chromosome made up of three genes, each one with three loci. As one can see, the number of twos in a boolean term determines its parsimony. Therefore, we will call the order of a boolean term [6] as the number of twos in its correspondent gene, so that higher order genes are the more parsimonious ones. The use of variable length representation in this problem is a natural choice, since one does not know, in principle, how many terms the minimal solution is made up of, i.e., the degree of simplification of the boolean function (Zebulum et al., 1997b).

We can compare our representation with the so-called gate level representation (Iba et al., 1997), which has been widely used in evolutionary electronics. In the gate level representation, the genotypes encode directly the digital circuitry, which is organized in logic gates (Zebulum et al., 1997a). The gate level representation hasn't yet been successfully applied to large combinational functions. In contrast, our approach promises to be suitable when the problem scales up. Nevertheless, our representation has two main disadvantages:

1. There is a particular class of boolean functions for which the decomposition in products connected by XOR logic (Miller et al., 1994) instead of OR will provide a more economic implementation. Our representation will not be efficient for this class of functions;

2. The gate level representation is more naturally suited to Field Programmable Gate Array (FPGA) applications, where the digital circuitry mapped by the genotype follows a particular class of FPGA architecture (Miller et al., 1997).

However, many important CAD tools based on conventional logic minimization techniques (Brayton et al., 1984), such as Espresso, represent digital circuits as a sum of product terms as we do in this work.

The number of fitness cases for combinational circuits increases exponentially with the number of inputs and, as a consequence, the hardness of the problem increases non-linearly as well. The authors have identified that, in general, as the number of inputs n surpasses the value of 6 or 7, special methodologies are required, such as inserting memory in the evolutionary system or carving up the genome space. Nonetheless, as it will be shown, the

[6]There is no relationship between our definition of the order of a boolean term and the conventional definition of schemata order.

structure of the target boolean function also plays a major role in the determination of the problem complexity.

The determination of the total genotype length n in this representation can be critical, depending on the target boolean function. If upper limit effects are observed after a particular choice of n, the value has naturally to be increased. However, this value can be fixed to the total number of minterms in the target boolean function specification, which provides the maximum size of a solution.

Evaluation

The fitness evaluation function takes correctness into account by measuring the number of hits according to the truth table. Additionally, we included a penalty term, as shown in Equation 5.

$$\text{Fitness} = Hits - k * penalty \tag{5}$$

There are three penalty sources in the evaluation function.

1. For each pair of repeated genes that a genotype may present, the penalty term is increased by one unit.
2. Genes that produce "1" outputs where the truth table is "0" also contribute to the penalty term. The reason is that, even though this kind of gene may present a high number of hits, it will never take part in the final solution (Hill and Peterson, 1981).
3. The penalty term is also increased for each gene that is out of the particular region to be sampled by the evolutionary algorithm (Equation 4) when the genome space is carved up.

The term k is a constant set by the user that determines the strength of the penalty term in the fitness evaluation function. Through much experimentation, the authors have found best results for values of k around 2^n, where n is the number of input variables. This can be explained by the fact that, in this case, both terms of the fitness function, hits and penalties, will have approximately the same range (the maximum number of hits is 2^n).

4.1.2 Applications

We show results for some basic boolean functions, such as multiplexers, comparators and parity functions.

The class of multiplexer combinational functions is one in which the ratio (number of terms of the optimal solution)/(number of minterms in the truth table) is very small, allowing a high degree of simplification. In this section, we examine the evolution of 6- and 11-input multiplexers.

The 6-input multiplexer is a standard combinational circuit commonly used for the assessment of evolutionary methodologies (Koza, 1992). This combinational function consists of 32 minterms, which can be simplified to a 4-term minimal solution. According to our representation, there are $3^6 = 729$ different genes; the task of the evolutionary algorithm is to find a genotype in which only the four genes representing the minimal terms are active. There are a total of ${C_{729}}^4 \cong 7.00 \times 10^{10}$ possible combinations of four genes in this genome space. Ruling out the task of also evolving the solution size, this number provides a good measure of the problem complexity.

Table 1: Survey on the 6-multiplexer evolution performance: percentage of successful executions, average size of the solutions in terms of number of gates, and the ratio between performance and size. The vary-length operator expression provides the rate at which it is applied to each mask bit; g is the generation index.

Methodology	Psucc(%)	Av. size	Psucc/Av. size	VaryLength
EA (1)	91	12.02	7.57	$0.2e^{-0.0005g}$
EA (2)	84	12.78	6.57	0.05
EA (3)	92	12.86	7.15	$0.2sin(\frac{g}{20})e^{-0.0005g}$
EA (4)	93	11.96	7.78	$0.2e^{-0.0005g}$
EA (5)	91	13.10	6.95	0.05
EA (6)	92	12.42	7.41	$0.2sin(\frac{g}{20})e^{-0.0005g}$
GP (1)	57	12.00	4.75	—
GP (2)	62	57.73	1.07	—

The results for the 6-input multiplexer are summarized in Table 1. It presents, for each VLR methodology, the percentage of success, the average solution size, and their ratio as a measure of performance. The vary-length operator expressions are also provided in this table. All the experiments have been carried out using: one-point crossover with a rate of 99%, a mutation operator with rate of 5% per genotype locus, a population of 30 individuals, exponential selection (c = 0.5 [7]), 300 generations, and 100 executions. Each execution lasts less then a minute in a Sparc Sun station model Ultra 2. Each individual can have a maximum of 20 active genes corresponding to a genotype of 20 x 6 = 120 loci. For comparison purposes, we have also included in Table 1 some results obtained in Blickle (1996), where explicit mechanisms to control the genotype size were used within the genetic programming methodology. Both our EAs and genetic programming (GP) process a similar number of individuals per execution in this comparison (9,000 in our EAs and 10,000 in GP). The following approaches are being compared in this table: EAs (1), (2), and (3) employ the ILG, UDIP, and OLG strategies respectively, without Pareto control; EAs (4), (5), and (6) employ the ILG, UDIP and OLG strategies, respectively, with Pareto control; GP (1) is the genetic programming methodology using deleting crossover with parsimonious pressure (Blickle, 1996), and GP (2) is the genetic programming methodology using deleting crossover without parsimonious pressure (Blickle, 1996).[8]

From this table, one can conclude that the evolutionary algorithm approaches developed by the authors outperform the genetic programming approaches, both in terms of performance and size of the solutions. We can also see that our three VLG methodologies tend to have a very similar performance, either with or without Pareto control. Using the ratio *Percentage of Success/Size* as a measure of the performance of the evolutionary systems, we can see that the ILG strategy slightly outperforms the other ones. None of the GP approaches use Automatic Defined Functions (ADFs) (Koza, 1994), though they include the operator IF in the functions' set, which is very useful for the particular case of the multiplexer problem, constituting the use of knowledge in the domain of this particular problem.

[7] See (Blickle,1996) for definition of the parameter c, which controls the selection pressure.

[8] To compare the size of the solutions, we estimated the number of gates necessary to implement the boolean terms obtained in our solutions and assumed that each GP tree node was equivalent to a logic gate.

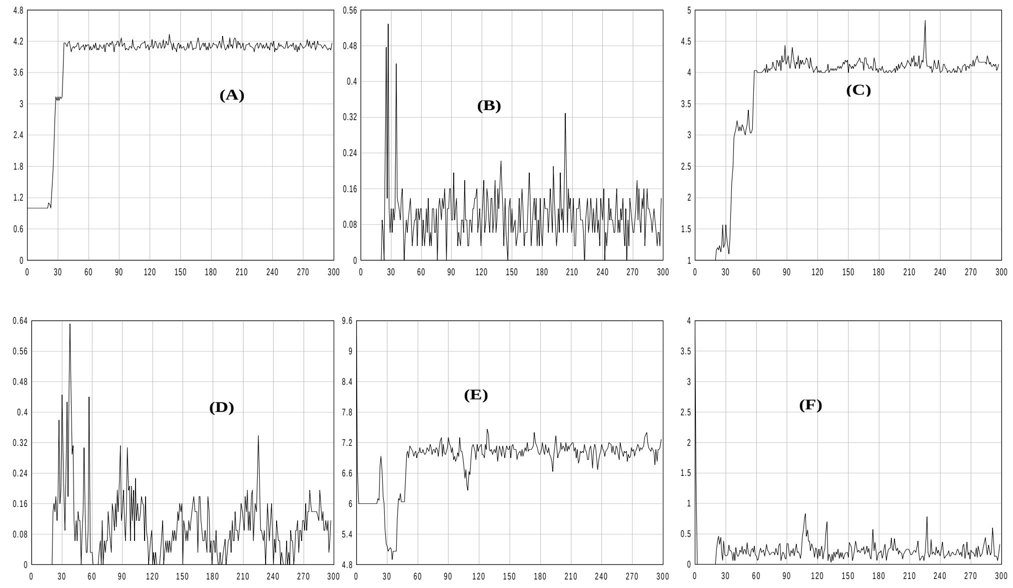

Figure 4: 6-multiplexer evolution using ILG (A, B), OLG (C, D), and UDIP (E, F) strategies. Axis X is the number of generations; axis Y gives the average genotype size measured in number of genes for graphs A, C, and E and the variance of the genotype size for graphs B, D, and F.

The vary-length operator expressions have been selected by assessing the EAs performance in many experiments. Particularly, this operator provides a way to control, under certain limits, the average parsimony and correctness of the final solutions.

The graphs in Figure 4 show the average behavior of the population in a typical execution for each of the three strategies employed. Graphs (A) and (B) show the average genotype length and variance of the genotype length, respectively, for the ILG strategy. Graphs (C) and (D) show the average genotype length and variance of the genotype length, respectively, for the OLG strategy. Graphs (E) and (F) show the average genotype length and variance of the genotype length, respectively, for the UDIP strategy.

We can observe that, in the ILG and OLG strategies (graphs A and C), the average size of the genotypes starts at the value of one gene and stabilizes with four genes–the optimum number of boolean terms for the 6-multiplexer. The variance of the genotypes' size along the evolutionary process is usually low, and the population stays homogeneous during evolution. On the other hand, in the UDIP strategy, the initial average genotypes' size is around 10 (Graph E), reflecting the fact that the initial individuals have their sizes randomly assigned between 1 and 20 genes. Also, in the first generation, due to the high selection pressure, there is a sharp fall in the average size of the genotypes and in the variance of the population (the initial values, 9.3 and 3.8 respectively, are not clear in Graphs E and F). This is due to the fact that, in the beginning, smaller individuals usually have higher fitness. It is interesting to notice that the low variance in the genotypes' length is one of the assumptions of the Species Adaptation Genetic Algorithm (Harvey, 1993). Also,

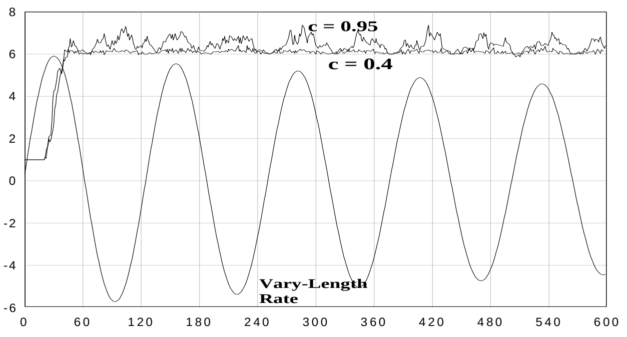

Figure 5: Effect of extreme degrees of selection pressure in the average genotype length for the 6-multiplexer using the OLG strategy. Axis Y gives the genotype size and axis X gives the generations. The vary-length operator rate (sine wave) is also shown in a different scale.

the population tends to remain homogeneous, thus avoiding problems of mating between radically different genotypes (Goldberg, 1989).

In order to evaluate the effect of the selection pressure, we ran two extreme cases for the 6-multiplexer: exponential selection with very low selection pressure (parameter c = 0.95) and exponential selection with very high selection pressure (parameter c = 0.4). The results, in terms of average genotypes' length, using the OLG strategy, are shown in the graph of Figure 5. It can be seen that, for high selection pressure, the average genotype length reaches an almost constant value quickly, while it tends to oscillate when lowering the selection pressure. Also shown in this graph is the vary-length operator rate (in a different scale), given by the slowly decaying sine wave defined in Table 1.

The synthesis of larger boolean functions, such as the 11-multiplexer, 8-comparator, and 8-parity can be accomplished by the introduction of enhancements to our basic methodology, such as the *partition of the genome space* and the use of the *memory paradigm*. In the former, we apply Equations 3 and 4, for $K = 3$; the latter consists of storing all the genes that do not add penalties to the fitness evaluation function (according to Equation 5), in the EA memory.

The 11-multiplexer is a combinational function defined by 1,024 minterms that are simplified to only 8 terms. There are $3^{11} = 177,147$ different genes according to our representation. The complexity of picking up the 8 minimal terms from the total number of genes can be estimated as ${C_{177,147}}^8 \cong 2.41x10^{37}$, therefore, 10^{27} times higher than the complexity of the 6-multiplexer. This statistic itself precludes the use of our previous EAs without any enhancement. We have carved up **uniformly** the genome space into five slices, thus using five parallel EAs. The memory paradigm and the ILG strategy have also been used. The genes found by each parallel EA may be taken to compose the overall solution, since they account for non-overlapping regions of the genome space, and there is no epistatic effect in our representation. Using this criteria, we have obtained **six** solutions over ten executions of our parallel EA; **four** of them included the eight optimal genes. The results compare very well with other references (Sakanashi et al., 1997), considering that we did not use any specific knowledge of the multiplexer functions in the design of the evolutionary

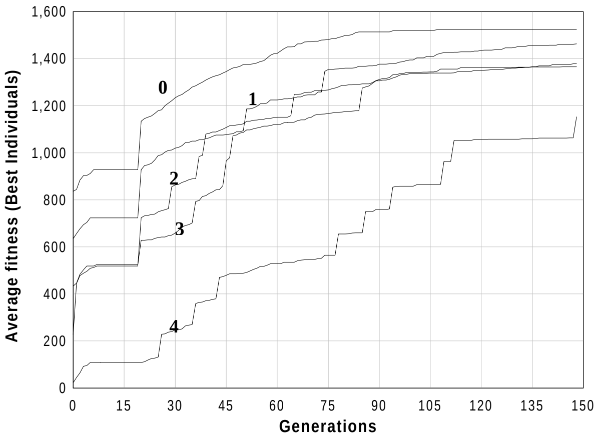

Figure 6: Average fitness in the 11-multiplexer experiment (EA number assigned to each curve).

algorithm. Furthermore, our methodology promises to be less time consuming then other methodologies applied to the problem. Each run processes a total of 22,500 individuals (30 individuals, 5 EAs, and 150 generations) against 36,000 (one solution found in 9 generations over a population of 4,000 individuals) in Koza (1994). Figure 6 shows the average fitness of the five EAs over ten executions. The best possible fitness is 2^{11} = 2,048 but is achieved by none of the processes since the solution is now distributed.

Table 2: Survey on the 8-comparator evolution: $(G)_{av}$ and $(F)_{av}$ are, respectively, the size and fitness of the best genotypes achieved, averaged over ten executions.

Methodology	$Psucc(\%)$	$(G)_{av}$	$(F)_{av}$
OLG with Memory	100	22.1	256 over 256
ILG with Memory	80	18.5	255.8 over 256
OLG without Memory	0	8.1	243.8 over 256

The effectiveness of the reuse of genetic material when initializing non-coding segments along evolution may be demonstrated by the synthesis of an 8-bit comparator. This device implements the logic function $A > B$, where A and B are 4-bit numbers, resulting in a total of eight inputs. This combinational function is specified by 120 minterms, which may be reduced to 15 terms using the standard CAD tool Espresso. Whereas the solution of the multiplexer functions are made up of high order terms (highly simplified terms), and the ones for the parity functions are made up of the lowest order terms (the own minterms), the solution to the 8-bit comparator includes both high order terms and minterms. There are a total of $3^8 = 6561$ possible genes. The complexity of gathering 15 terms can be calculated by ${C_{6561}}^{15} \cong= 10^{45}$. Table 2 shows the average performance of our methodology when using the OLG and the ILG strategies, both with the memory paradigm, and the OLG strategy without the memory paradigm. As it can be seen, the reuse of genetic material in non-coding regions produces an extraordinary benefit to the performance in this case. It can also be observed that the ILG strategy arrived at more parsimonious solutions in terms of number of genes; the best solution found presented a total of 16 genes, one gene more than the optimal solution. The strategies employing the memory paradigm took around four hours to run 10 executions in a SPARC classic workstation.

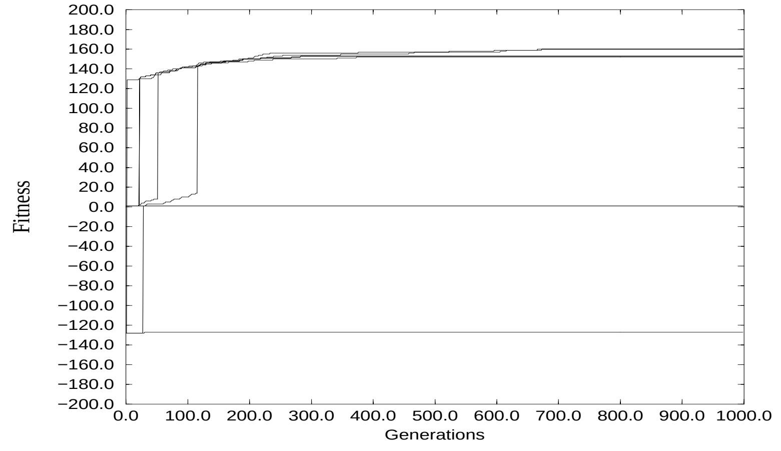

Figure 7: Fitness of the best genotype along the evolutionary process for each of the ten running EAs of the 8-parity problem (1 execution).

The 8-parity combinational function is specified by 128 non-simplifiable minterms, leading to a complexity of ${C_{6561}}^{128} \cong 10^{271}$. The genome space has been divided into ten regions, one EA has been used to sample each region. Figure 7 shows the best individual fitness for each running evolutionary algorithm; the ones that sample regions with more minterms reach higher fitness values. At the end, all 128 minterms have been found. Nevertheless, this is an example for which our representation is not efficient, since parity functions can be more easily expressed using the exclusive OR operator.

The representation used in this case study solves some of the problems of the *gate level representation* concerning the evolution of larger digital circuits. However, in order to make this tool competitive with conventional ones, such as Espresso, our methodology must be further improved to handle digital circuits larger than the ones presented in this paper. The authors see the process of carving up the genome space as the most promising to overcome this problem. Additionally, promising gate level approaches for the evolution of digital circuits, which can compete with the state of art in CAD, have been recently reported (Miller at al., 1997; Zebulum et al., 1998c).

4.2 Analog Circuits

Analog circuit design benefits more from the use of artificial evolutionary systems than digital design, since CAD tools are not as efficient for analog design (Johns and Martin, 1997). We present results on passive filters and transistor based amplifiers.

Throughout this section, we show our results using a performance measure that rates fitness and parsimony as given by:

$$r_{best,j} = \frac{U_T(F_{best,j}).F_{best,j}}{G_{best,j}} \tag{6}$$

$$U_T(f) = 1,\ if\ f > T \quad , \quad U_T(f) = 0\ otherwise \tag{7}$$

where $F_{best,j}$ and $G_{best,j}$ stand, respectively, for the fitness and genotype size of the best individual for a particular execution j of the evolutionary algorithm. $U_T(f)$ is a unit step function over a fitness threshold T selected by the user. If the achieved fitness is better than the specified threshold, the ratio defined above will be greater than 0, so that the higher its value, the better the methodology. When taking into account a set of executions, two other measurements can be derived: $(r_{best})_{max}$ and $(r_{best})_{av}$ that correspond to the maximum and average values of $r_{best,j}$ over a number of executions of the EA.

The main idea behind these indexes is to give the same emphasis to the circuit's fitness and size. There can be different versions for the r-indexes. For instance, if the circuit correctness is more important than its parsimony, the fitness value in the above equation could be squared.

4.2.1 Passive Filters

A filter is a circuit which alters the characteristic of a periodic electronic signal (Bishop, 1991). If the frequency of this input signal lies in the filter *passing band*, then it must appear in the circuit output without attenuation; if the input signal frequency lies in the filter *stop band*, then it must be completely rejected. Particularly, passive filters are made up of resistors, capacitors, and inductors.

The use of artificial evolutionary systems in the design of passive filters has already been studied through genetic algorithms (Grimbleby, 1995) and genetic programming (Koza et al., 1996). This case study serves a twofold purpose: performing a theoretical investigation of the application of VLR systems in filter design and assessing the practical suitability of using evolutionary systems as a CAD tool for analog design. As this application is oriented towards the conception of practical circuits, the EA uses only electric components with standard values (Horrocks and Spittle, 1996). This section extends the results shown in Zebulum et al. (1998a).

In our chromosome representation, each gene encodes a particular electric component by specifying its nature (resistor, capacitor, or inductor), its value from a set of preferred values (Horrocks and Spittle, 1996), and its connecting points. Any arrangement of components is allowed, which permits the exploration of new topologies. Based on previous work (Koza et al., 1996), we have devised the following fitness evaluation function:

$$\text{fitness} = 1,000 - \sum_{i=1}^{n} w_i |Target_i - Output_i| \tag{8}$$

where $Target_i$ is the desired frequency response, $Output_i$ is the obtained response, and n is the total number of output samples. According to Equation 8, the fitness is defined by the weighted sum of the deviations between the actual output and the desired filter response; this sum is multiplied by -1 and added to a constant chosen to be 1,000. The weighted sum is multiplied by -1 because the lower its value, the higher the fitness should be; the added constant has been a way to assign to fully compliant circuits a maximum defined score of 1,000 points, and it is just used for the sake of visualization. The weight w_i should take a maximum value for frequency points inside the passband, an intermediate value for frequency points inside the stop band, and a minimum value for other frequency points. They assume the values of 20, 10, and 1 for these three cases respectively. Although this fitness equation worked well, setting the values of the weights has been an interactive and time consuming process. To evaluate the circuits, we have used the SMASH simulator.

In order to demonstrate the flexibility of our methodology, we present results for the design of low-pass, band-pass, and notch filters. In all cases, we have used VLR methodologies with genotypes made up of 20 genes corresponding to 20 components, exponential selection (c = 0.7), one point crossover with 80% rate, and 400 generations.

Table 3: Results on analog filters evolution. The first column refers to the particular experiment (E): 1, 2, and 3 refer to the ILG strategy applied to low-pass, band-pass, and notch filters respectively; 4, 5, and 6 refer to the UDIP strategy applied to low-pass, band-pass, and notch filters respectively; 7, 8, and 9 refer to the OLG strategy applied to low-pass, band-pass, and notch filters, respectively. UDIP and OLG strategies employ the Pareto control technique. P_M is the mutation percentage; the vary-length operator expression provides the rate at which it is applied for each mask bit; g is the generation index.

E	$(r_{best})_{max}$	$(r_{best})_{av}$	F_{best}	G_{best}	$\%suc$	P_M	VaryLength
1	228.6	67.4	914.4	4	70%	0.8%	$0.2e^{-0.005g}$
2	110.3	18.9	882.3	8	30%	2%	$0.2e^{-0.0005g}$
3	73.1	26.3	803.9	11	45%	2%	$0.2e^{-0.001g}$
4	94.6	70.1	946.2	10	95%	0.8%	0.03
5	77.8	35.7	933.9	12	50%	2%	0.03
6	67.6	13.2	878.5	13	20%	2%	0.03
7	121.5	74.4	972	8	80%	2%	$0.2sin(\frac{g}{20})e^{-0.0005g}$
8	164.1	23	820.6	5	20%	2%	$0.2sin(\frac{g}{20})e^{-0.0005g}$
9	108.2	8.3	865.7	8	10%	2%	$0.2sin(\frac{g}{20})e^{-0.0005g}$

Table 3 summarizes the results in terms of the performance measures defined in Equations 6 and 7. Also displayed in this table are the fitness (F_{best}) and genotype size (G_{best}) values for the individual that presented the highest $fitness/parsimony$ ratio over the experiments. The percentage of success ($\%suc$) is also given in this table. This is defined by the percentage of EA executions in which the minimum fitness threshold (see Equations 6 and 7) has been achieved. This threshold was set to 900 out of 1,000 for the low-pass filter and 800 out of 1,000 for the band-pass and notch filters. Each of the nine experiments consisted of 20 executions of the evolutionary algorithm. Each experiment lasts around 16 hours in a Sun Ultra Enterprise 2 server with one 300MHz Ultra Sparc processor.

In terms of the r-indexes, one can see from Table 3 that the ILG and OLG strategies achieve the highest $(r_{best})_{max}$ values, whereas the UDIP strategy is slightly better regarding the average statistics $(r_{best})_{av}$. Furthermore, the percentage of successful experiments for the OLG strategy, in the case of band-pass and notch filters, is much lower than in the case of the ILG and UDIP strategies. This table also presents the mutation rate and the general expression of the vary-length operator in each experiment. We observed that the EA performance is very sensitive to these two parameters. The authors have performed comparative tests to find good values for these parameters.

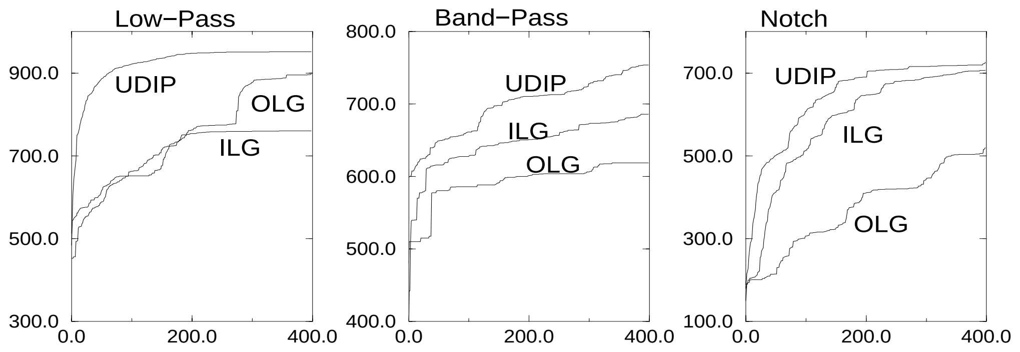

Figure 8: Comparison among the performance of the VLR strategies in terms of fitness only: best genotypes average fitness for the low-pass, band-pass, and notch filters over twenty executions – axis Y is the average fitness and axis X represents the generations.

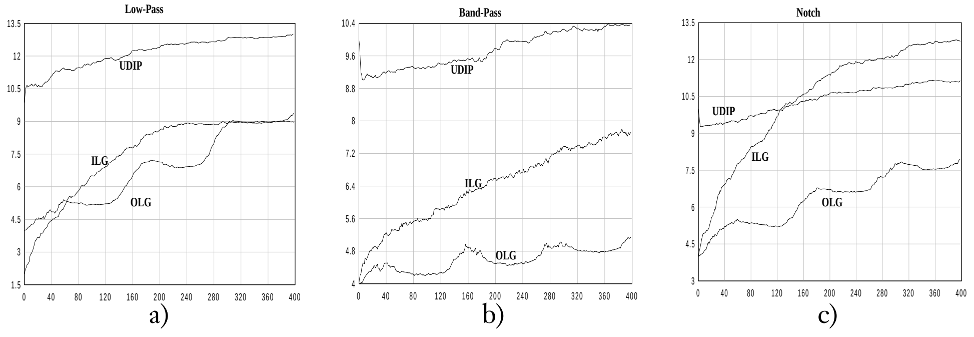

Figure 9: Comparison among the performance of the VLR strategies in terms of average genotypes' size only: average genotypes' size for the low-pass filter, band-pass, and notch filters over twenty executions – axis Y is the average genotype size expressed in number of electronic components and axis X represents the generations.

The graphs of Figure 8 show the best individual fitness along the evolutionary process, averaged over the 20 executions for the nine experiments. From these graphs we can see that the UDIP strategy usually achieves best fitness values, and the OLG strategy performance deteriorates for more complex filters' specifications, such as band-pass and notch filters.

Figure 9 depicts the genotype length averaged over the 20 executions for the nine experiments. The graphs show that, while the ILG strategy produces a constant trend towards growth, the OLG strategy tends to oscillate, and the UDIP strategy shows just a slight tendency towards genotype growth.

The low-pass filter is specified to have a passing band below 1,000 Hz and a stop band above 2,000 Hz (Koza et al., 1996). The band-pass filter has been specified to accept frequencies between 2,000 and 3,000 Hz and reject the ones below 1,000 Hz and above 4,000 Hz. The notch filter has been specified to accept frequencies below 1,000 Hz and above 5,000 Hz, rejecting the ones between 2,000 and 4,000 Hz. Figures 10, 11, and 12 show the schematics of the best low-pass (OLG strategy, fitness of 981 out of 1,000), band-pass (UDIP strategy, fitness of 955 out of 1,000), and notch (ILG strategy, fitness of 909 out of 1,000) filters obtained in these experiments. Figure 13 shows their respective frequency responses. These circuits have been the best obtained considering their fitness only, i.e., not taking into account their performance according to their r-indexes.

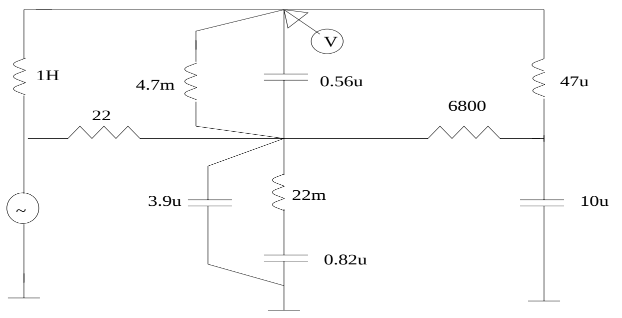

Figure 10: Best low-pass filter evolved considering fitness only (using the OLG strategy). The V marker indicates the output.

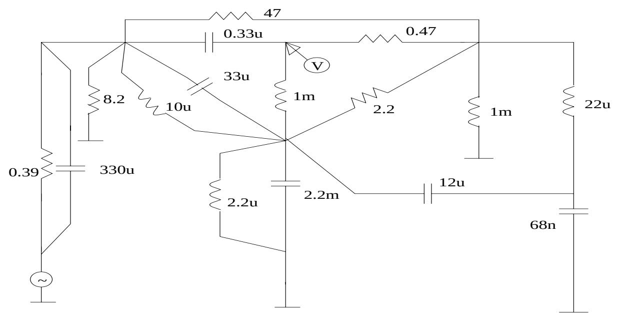

Figure 11: Best band-pass filter evolved considering fitness only (using the UDIP strategy). The V marker indicates the output.

Let us compare our results for the evolution of the low-pass filter (also called "Brick Wall" filter) with the genetic programming methodology (Koza et al., 1996). While a fully compliant circuit has been evolved (Expression 8 equal to 1,000) through genetic programming, using our VLR methodology and the ILG strategy, we arrived very close to the fully compliant circuit (fitness of 981 out of 1,000). Nevertheless, our VLR methodology evolved a more parsimonious circuit (10 components against 16 in the GP methodology). Additionally, the total number of individuals processed within our framework has been much smaller than the one in GP: 320,000 (40 individuals, 400 generations, and 20 executions) in our methodology against 10,240,000 (320,000 individuals and 32 generations) in GP. The fact that we used only standard components values also contributed to increasing the hardness of the task.

The three filters, shown in the figures, presented a flat response in their passing bands, which is a desirable feature. The low pass filter presented a maximum attenuation of -0.65dB

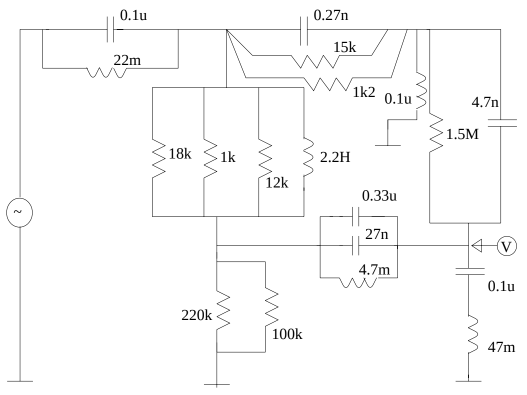

Figure 12: Best notch filter evolved considering fitness only (using the ILG strategy). The *V* marker indicates the output.

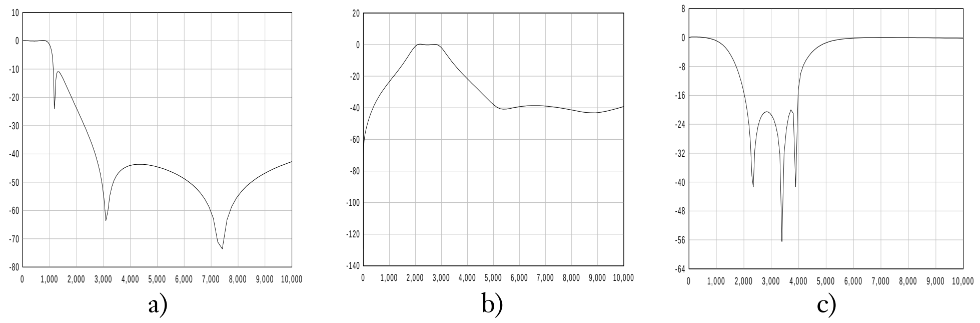

Figure 13: Frequency responses of the passive filters: Y axis is the output voltage in decibels and X axis is the frequency in Hertz. (a) low-pass, (b) band-pass, (c) notch.

in its passing band and a steepness of 24.2 dB/1000Hz in its transition band. The band-pass filter presented a maximum attenuation of -1.19dB in its passing band and steepness of 20.5dB/1000Hz and 22.5dB/1000Hz in its transition bands. The notch filter presented a maximum attenuation of -1.29dB in its passing band and steepness of 30.4dB/1000Hz and 17.4dB/1000Hz in its transition bands. The filters' steepness can be improved, at an expense of its passing band flatness, by changing the weights of the fitness evaluation function. For instance, *we could achieve a steepness of **55dB/1000Hz** for the low-pass filter by changing these weights, but the maximum passing band attenuation deteriorated to -2dB.*

The Use of the Memory Paradigm

The EAs used in the analog circuits case studies presented so far have not employed the memory paradigm. In order to use this paradigm, we changed our representation so that each gene now encodes for a circuit mesh whose performance as a filter can be easily assessed. The filter topologies are, therefore, constrained to circuits with a variable number of parallel meshes (Zebulum et al., 1998a). We made experiments for the particular case of the band-pass filter, defined above, using the ILG strategy. During evolution, genes encoding for meshes behaving as either low-pass filters or high-pass filters were stored

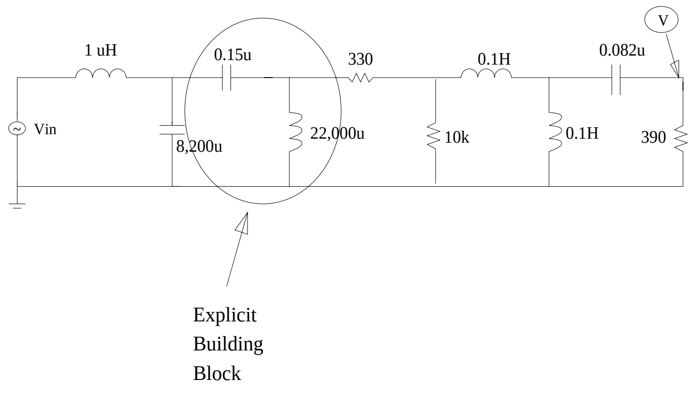

Figure 14: Evolved band-pass filter using the memory paradigm (fitness of 862 out of 1,000) and explicit building block found in memory.

Table 4: Results on the use of the memory paradigm in the evolution of a band-pass filter over 10 executions: case 1 uses the memory paradigm and mesh representation; case 2 refers to the use of the mesh representation without the memory paradigm.

Case	$(r_{best})_{max}$	$(r_{best})_{av}$	F_{best}	G_{best}	$\%suc$	$(F)_{av}$
1	103.9	27.1	830.9	8	30%	711.2
2	83.0	14.9	830.2	10	18.2%	613.9

in the genetic memory, since these kinds of circuits can be useful building blocks for the evolution of the band-pass filter. Figure 14 shows the best band-pass filter evolved using the memory paradigm. The authors identified one explicit building block found both in the memory and in the best circuit. Figure 15 shows the frequency response of the whole circuit and of the explicit building block (a high-pass filter with cut-off frequency below 100 Hz). Table 4 compares the following results: constraining the circuit topologies to meshes using the memory paradigm (over 10 executions) and constraining the circuit topologies to meshes without using the memory paradigm (over 10 executions). Besides the statistics shown in Table 3, we added the $(F)_{av}$ statistics to Table 4, which accounts for the final best fitness value averaged over 10 executions.

The results suggest that the memory paradigm improves the performance; though, opposing to their digital counterparts in which there is no epistasis effect among genes, in this case, the individual meshes will affect each other to a certain extent due to electric coupling.

4.2.2 Transistor Based Amplifiers

The design of analog amplifiers is usually a complex task and can be considered more a form of art than science, requiring experience and intuition from the designer (Johns and Martin, 1997). Koza et al. (1996) have published pioneering applications, applying genetic programming to the design of amplifiers. We have used our VLR methodologies to tackle this problem. The same chromosome representation employed in the passive filters

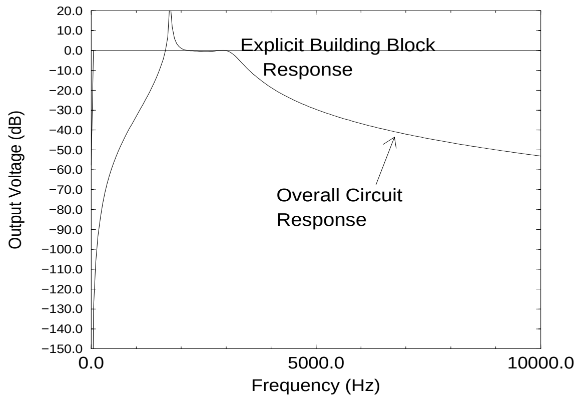

Figure 15: Frequency responses for the circuits shown in Figure 14.

application has been used; each gene may encode a npn transistor, a pnp transistor, or a resistor. The gene also encodes the components' connecting points and their magnitude, in the case of a resistor. The fitness evaluation function is made through the evaluation of the DC response of the circuit, as suggested in Bennett et al. (1997). The SMASH circuit simulator has also been used in this case study, and the final evolved circuits have also been simulated using SPICE. After applying an input signal of the order of mV, we measure the induced variation in the output voltage. The higher the output variation, the better the amplifier.

Table 5: Results on analog amplifier's evolution over 10 executions: fitness is given in volts and genotype size is given in number of electronic components – UDIP and OLG strategies employ the Pareto control technique.

Methodology	$(r_{best})_{max}$	$(r_{best})_{av}$	F_{best}	G_{best}	$\%suc$	$(F)_{av}$
ILG	1.04	0.23	11.41	11	50%	2.61
UDIP	0.31	0.10	2.47	8	80%	0.76
OLG	1.22	0.31	6.08	5	70%	1.97

Table 5 compares the strategies' performances; the minimum specified fitness threshold T has been set to 20 dB gain amplifiers. Although the OLG strategy presented best r-indexes, the ILG strategy evolved the highest gain amplifier (see F_{best}).

Figure 16 shows the schematic of a 55 dB gain amplifier obtained by our methodology using the ILG strategy; Figure 17 shows its DC response to a 20 mV input signal. This figure shows both the response obtained in simulation and the behavior of the circuit when **implemented in reality**. As can be seen, the simulated response is steeper than the measured response, although both curves correspond to an amplifier. The difference between the responses may be due to the following:

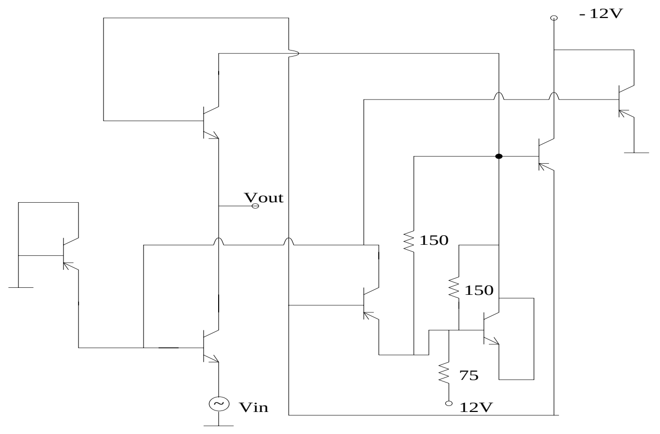

Figure 16: Schematic of the best amplifier obtained by the ILG strategy.

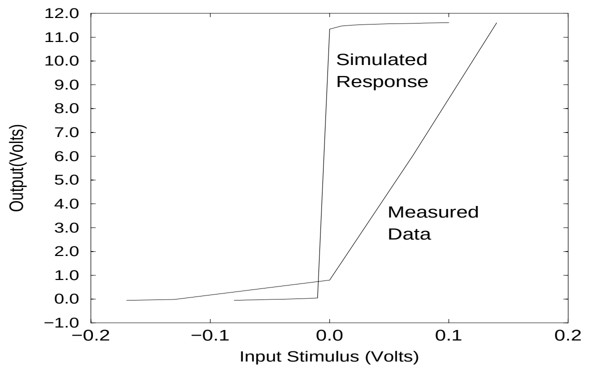

Figure 17: Comparison between the DC responses of the circuit in Figure 16 in simulation and in real implementation.

1. The behavior of bipolar transistor devices may vary slightly among one another due to small differences in their parameters.

2. There are minor differences between the transistors' parameters used in simulation and the ones of the real transistors.

3. Finally, in the case of evolved circuits, the unconventional circuit topologies yielded may be not accurately modeled by simulators, since many transistors now operate in regions that are not commonly used in conventional design, such as the *reverse region* (Laker and Sansen, 1994); in the case of the circuit in Figure 16, the input signal is being supplied to the emitter of a particular transistor – a procedure that is not normally followed conventionally.

A more complete investigation of the above issue can be found in Zebulum et al. (1998b).

A similar attempt to evolve an amplifier is reported in Bennett et al. (1997). Using the genetic programming methodology and the SPICE simulator, they achieved a 60 dB gain circuit with 22 transistors, while we arrived at a 55 dB gain circuit (measured in simulation) with 7 transistors. In Bennett et al. (1997), linearity has also been taken into account in the fitness function. We have processed a total of 80,000 individuals (40 individuals, 200 generations, and 10 executions, each one taking around 4 hours in a Sparc Ultra 2) against 69,760,000 in GP (109 generations and 640,000 individuals). We also ensured the testability of our circuits by mounting them in protoboards, an issue that increases the problem complexity (Zebulum et al., 1998b).

5 Biological Analogy

This section contrasts natural mechanisms with the artificial evolutionary strategies proposed in this work. The human genome consists of two types of DNA, coding and non-coding. The former is effectively transcribed to produce proteins or regulate their production (Ridley, 1996). On the other hand, DNA that does not code for genes is called non-coding DNA or introns. One evolutionary hypothesis says that non-coding sequences are $selfish$ DNA, contributing nothing to the well-being of the organism. A non-coding sequence may be either active or passive and may have evolved from mechanisms such as unequal crossover (Ridley, 1996). The main idea is that these facts contribute to make the genotype-phenotype mapping performed by nature far different from the one employed in traditional genetic algorithm models.

Therefore, variable length representation evolutionary systems are more plausible from a biological point of view. Concerning the variable length representation in biology, geneticists still disagree about the proportion of non-coding DNA in the human genome (Ridley, 1996)–estimates made by DNA re-association experiments vary from 10% to 50%. The human genome contains around 100,000 genes, meaning that human beings use about 100,000 different kinds of proteins in their lives. As a protein is made up of an average of 300 amino-acids, which leads to about 1,000 nucleotides, one concludes that there are around 10^8 coding bases in the human genome. Notwithstanding, a human cell contains about 3 x 10^9 nucleotide units, much more than the amount needed to code for proteins. In spite of the imprecision, it is certain that most DNA does not code for genes, being called non-coding DNA.

Our model is, obviously, a great simplification of natural mechanisms with a major difference: macroevolutionary changes in nature are carried out through adaptation and gradual changes (Ridley, 1996) and not through genome expansion, as defined in our model. This fact suggests that the engineering benefits of VLG may be completely different from any advantage it might confer to nature. Nonetheless, our artificial system borrows some principles from nature that are not seen in standard EAs: the existence of both coding and non-coding genes, increasing growth in phenotype complexity, the accessibility of the effect of each gene in the phenotype, and a higher degree of preservation of useful genetic material.

6 Conclusions

This work has demonstrated the high level of correlation between two areas of study: evolutionary electronics and variable length representation evolutionary algorithms. The latter can get benefits from the large source of case studies provided by the former; the former, on the other hand, may produce many results of practical importance for CAD development by borrowing concepts from the latter.

We have made an extensive study of cases, including analog and digital electronics, covering most of the evolutionary electronics applications being investigated at the present moment. In our experiments, we have compared the performance of different strategies for sweeping the VLR genome space, investigated implicit and explicit mechanisms for balancing accuracy and parsimony, and proposed an explicit way of using non-coding segments.

Our experiments showed that the increasing length genotypes strategy for sweeping the genome space produced better results for many of the applications developed here. The implicit mechanism for controlling the solutions' size is one of the more interesting features of the ILG strategy.

A technique based on the Pareto concept for multi-objective optimization has been applied as an explicit way of balancing parsimony and accuracy. The main effect of using this mechanism is to indirectly increase the selection pressure towards parsimonious and fit solutions.

The use of the memory paradigm allowed the exploration of non-coding segments and has produced significant benefits in the evolution of large combinational functions. Even though this kind of non-coding material is usually seen as a problem in genetic programming, it is well suited to the application of the technique developed by the authors. The main challenge to getting the benefits of the memory paradigm in a broader domain of applications is finding representations with a low level of epistasis among genes. The authors are now investigating the application of this concept to the problem of VLSI placement.

Regarding future work, the following topics are foreseen: extension of the applications on combinational functions evolution to handle circuits of larger sizes, further investigation on the use of the memory paradigm on analog circuits evolution, and producing professional CAD tools for analog circuit synthesis using evolutionary algorithms.

Acknowledgments

The authors wish to thank CAPES Brazilian Federal Agency for the financial support and Dr. Adrian Thompson, Dr. Inman Harvey, and Dr. David Nicholson from CCNR, Sussex, UK, for the collaboration.

References

Bäck, T. (1994). Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. In Fogel, D., chair, *Proceedings of the International Conference on Evolutionary Computation*, pages 57–62, IEEE Press, Piscataway, New Jersey.

Banzhaf, W., Nordin, P., Keller, R. and Francone, F. (1998). On some emergent properties of variable size evolutionary algorithms. *Genetic Programming – An Introduction*. Morgan Kaufmann, San Francisco, California.

Bennett III, F. H., Koza, J. R., Andre, D. and Keane, M. A. (1997). Evolution of a 60 Decibel Op Amp Using Genetic Programming. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 455–469, Springer-Verlag, Berlin, Germany.

Bishop, O. (1991). *Practical Electronic Filters*. Babani Electronics Books, London, England.

Blickle, T. (1996). *Theory of Evolutionary Algorithms and Application to System Synthesis*. Ph.D. thesis, Department of Electrical Engineering, Swiss Federal Institute of Technology, Zurich, Switzerland.

Brayton, R. K., Hatchel, G. D., McMullen, C. T. and Sangiovanni-Vincentelli, A. L. (1984). *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer, Boston, Massachusetts.

Esbensen, H. (1994). A Macro-cell global router based on two genetic algorithms. In *Proceedings of the European Design Automation Conference*, pages 428–433, IEEE Computer Society Press, Los Alamitos, California.

Esbensen, H. and Mazumder, P. (1994). SAGA: A Unification of the genetic algorithm with simulated annealing and its application to macro-cell placement. In *Proceedings of the VLSI Design Conference*, pages 211–214, IEEE Computer Society Press, Los Alamitos, California.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16.

Gero, J. S. (1998). New Models for Evolutionary Designing. In Bentley, P., chair, *Proceedings of the Workshop on Evolutionary Design*, pages 37–41, University College London Press, London, England.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G., editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, San Mateo, California.

Goldberg, D. E., Deb, K. and Korb, D. (1990). An Investigation of Messy Genetic Algorithms. Technical Report TCGA 90005, TCGA, University of Alabama, Tuscaloosa, Alabama.

Grimbleby, J. B. (1995). Automatic Analogue Network Synthesis Using Genetic Algorithms. In *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pages 53–58, IEE Conference Publication No. 414, London, England.

Harvey, I. (1993). *The Artificial Evolution of Adaptive Behaviour*. D.Phil. thesis, School of Cognitive and Computing Science, University of Sussex, Brighton, Sussex, England.

Harvey, I. and Thompson, A. (1997). Through the Labyrinth Evolution Finds a Way: A Silicon Ridge. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 406–422, Springer-Verlag, Berlin, Germany.

Hill, F. J. and Peterson, J. R. (1981). *Introduction to Switching Theory and Logical Design*. John Wiley and Sons, New York, New York.

Horrocks, D. H. and Spittle, M. C. (1996). Component Value Selection For Activate Filters Using Genetic Algorithms. In Hirst, T., chair, *1st On-line Workshop on Soft Computing (WSC1)*, Special Session on Evolutionary Electronics.

Iba, H., Iwata, M. and Higuchi, T. (1997). Machine Learning Approach to Gate-Level Evolvable Hardware. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 327–343, Springer-Verlag, Berlin, Germany.

Johns, D. A. and Martin, K. (1997). *Analog Integrated Circuit Design*. John Wiley and Sons, New York, New York.

Kommu, V. and Pomenraz, I. (1993). GAFAP: Genetic Algorithm for FPGA technology mapping. In *European Design Automation Conference*, pages 300–305, IEEE Computer Society Press, Los Alamitos, California.

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

Koza, J. R. (1994). *Genetic Programming II Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, Massachusetts.

Koza J. R., Bennett III, F. H., Andre, D. and Keane, M. A. (1996). Four Problems for which a Computer Problem Evolved By Genetic Programming is Competitive with Human Performance. In Bäck, T., Kitano. H. and Michalewicz, Z., program committee, *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pages 1–10, IEEE Press, Piscataway, New Jersey.

Kruiskamp, W. and Leenaerts, D. (1995). DARWIN: CMOS opamp synthesis by means of genetic algorithm. In *Proceedings of the 32nd Design Automation Conference*, pages 698–703, ACM Press, New York, New York.

Laker, K. R. and Sansen, W. (1994). *Design of Analog Integrated Circuits and Systems*. McGraw-Hill, Singapore.

Langdon, W. (1997). Fitness causes bloat in variable size representations. In O'Reilly, U.-M., editor, *ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation*, Michigan State University, East Lansing, Michigan.

Layzell, P. (1998). A New Research Tool for Intrinsic Hardware Evolution. In Sipper, M., Mange, D. and Pres-Uribe, A., editors, *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware(ICES98)*, pages 47–56, Springer Verlag, Berlin, Germany.

Miller, J. F. and Thomson, P. (1995). Combinational and Sequential Logic Optimization Using Genetic Algorithms. In *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pages 34–38, IEE Conference Publications No. 414, London, England.

Miller, J. F., Thomson, P. and Fogarty, T. (1997). Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. In Quagliarella, D., Periaux, J., Poloni, C. and Winter, G., editors, *Genetic Algorithms Recent Advancements and Industrial Applications*, chapter 6. Wiley, New York, New York.

Miller, J. F., Luchian, H., Bradbeer, P. V. G. and Barclay, P. J. (1994). Using a Genetic Algorithm for Optimising Fixed Polarity Reed-Muller Expansions of Boolean Functions. *International Journal of Electronics*, 76:601–609.

Nordin, P., Francone, F. and Banzhaf, W. (1995). Explicitly Defined Introns and Destructive Crossover in Genetic Programming. In Angeline, P. and Kinnear, K., editors, *Advances in Genetic Programming II*, pages 111–134. MIT Press, Cambridge, Massachusetts.

Ridley, M. (1996). *Evolution*. Second Edition. Blackwell Science, Cambridge, Massachusetts.

Rogenmoser, R., Kaeslin, H. and Blickle, T. (1996). Stochastic Methods for Transistor Size Optimization of CMOS VLSI Circuits. In Voigt, H. M., Ebeling, W., Rechenberg, I. and Schwefel, H.-P., editors, *Proceedings of Parallel Problem Solving from Nature IV*, pages 849–858, Springer-Verlag, Berlin, Germany.

Sakanashi, H., Higuchi, T., Iba, H. and Kahazu, Y. (1997). Evolution of Binary Decision Diagrams for Digital Circuit Design Using Genetic Programming. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 470–481, Springer-Verlag, Berlin, Germany.

Sipper, M. (1997). Report of the Working Group in Evolutionary Electronics (EvoElec). Miller, J., chair, *EvoNet - The Network of Excellence in Evolutionary Computation*, University of Napier, Scotland, UK.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. Ph.D. thesis, Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania.

Thompson, A. (1996). *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution*. D.Phil. thesis, University of Sussex, Brighton, Sussex, England.

Thompson, A. (1997). An evolved circuit, intrinsic in silicon, entwined with physics. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 390–405, Springer-Verlag, Berlin, Germany.

Thompson, A., Harvey, I. and Husbands, P. (1996). Unconstrained Evolution and Hard Consequences. In Sanchez, E. and Tomassini, M., editors, *Towards Evolvable Hardware: An International Workshop*, pages 136–165, Springer-Verlag, Berlin, Germany.

Wu, A. S. and Lindsay, R. K. (1995). Empirical studies of the Genetic Algorithm with Noncoding segments. *Evolutionary Computation*, 3(2):121–147.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M. (1997a). Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications. In Higuchi, T. and Iwata, M., editors, *Proceedings of the First International Conference on Evolvable Systems (ICES96)*, pages 344–358, Springer-Verlag, Berlin, Germany.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M. (1997b). Increasing Length Genotypes in Evolutionary Electronics. In O'Reilly, U.-M., chair, *ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation*, Michigan State University, East Lansing, Michigan.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M. (1998a). Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design. In Fogel, D., chair, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 434–439, IEEE Press, Piscataway, New Jersey.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M.(1998b). Analog circuits evolution in extrinsic and intrinsic modes. In Sipper, M., Mange, D. and Pres-Uribe, A., editors, *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98)*, pages 154–165, Springer-Verlag, Berlin, Germany.

Zebulum, R. S., Pacheco, M. A. and Vellasco, M. (1998c). Evolutionary systems applied to the synthesis of a CPU controller. In Yao, X., chair, *Proceedings of the the Second Asia Pacific Conference on Simulated Evolution and Learning, SEAL98*, Session 17, Volume 2, University of New South Wales, Sidney, Australia.

Zhang, B. (1997). A taxonomy of control schemes for genetic code growth. In O'Reilly, U.-M., chair, *ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation*, Michigan State University, East Lansing, Michigan.

Zhang, B. and Mühlenbein, H. (1995). Balancing Accuracy and Parsimony in Genetic Programming. *Evolutionary Computation*, 3(1):17–38.