

Automatic Design of Electronic Amplifiers using Grammatical Evolution

Federico Castejón and Enrique J. Carmona

Dpto. de Inteligencia Artificial, ETSI Informática, Universidad Nacional de Educación a Distancia (UNED), Juan del Rosal 16, 28040, Madrid, Spain

fcastejon@gmail.com

ecarmona@dia.uned.es

Abstract. An algorithm for automatic synthesis of one stage analog amplifiers based on Grammatical Evolution (GE) and Koza's developmental expressions is presented. GE is an evolutionary algorithm which can generate code in any programming language and uses variable length binary strings. The binary genome determines which production rules in a BNF grammar definition are used in a genotype to phenotype mapping process to a program. Koza introduced developmental functions in Genetic Programming for the synthesis of analog circuits. Our GE-based algorithm uses a grammar comprising only ten of these functions. The fitness function has different terms to meet the different design specifications of the amplifier. Finally, this methodology was applied to a case study and the best circuit obtained proved to be a good approximation to the target amplifier. Also a comparison to a human designed amplifier is shown.

Keywords: Automatic Design of Electronic Circuits, Evolutionary Algorithm, Grammatical Evolution, Developmental expressions, NGSpice.

1 Introduction

Since late 70s, digital circuits have been replacing analog circuits in general, but some functions still have to remain analog mostly because transducers are analog as well. Unlike digital circuits, analog circuits still require to be designed by experts due to the lack of general analog design tools. Much effort has been done in Electronic Design Automation (EDA) to get aiding tools in the design of analog or mixed signal integrated circuits, like for example standard cell libraries or rule based expert systems [2].

Automatic circuit synthesis comprises two tasks: topology selection and circuit sizing. These tasks can be accomplished together or separately [5]. In this field, evolutionary algorithms have been initially used for circuit sizing among other optimizing algorithms like simulated annealing. On the other hand, topology has been selected by human designers, or aided by formerly mentioned EDA tools.

The field of Evolutionary Electronics appeared in 1998 [11], and its goal is the synthesis of electronic circuits using evolutionary algorithms. Inspired in natural selection, evolutionary algorithms can obtain solutions to complex problems using trial and error. Thus, using a tentative population of potential solutions, they are selected according to their fitness or closeness to the desired solution. Then the best solutions are crossed over and mutated. Successive generations lead to survival of the fittest. Evolutionary algorithms applied to synthesis tasks do not use design rules nor expert knowledge for the design [3], but can lead to unconventional solutions which challenge human designers intuition [1]. On the other hand, obtained circuits can be difficult to analyze by human designers or lead to rejection.

Apart from the design of integrated and digital circuits, evolutionary algorithms have been used with good results for automatic synthesis of analog circuits [9,10]. In particular, Koza and collaborators' work in this area, using genetic programming, is especially relevant [4].

We present an algorithm for synthesizing one stage analog amplifier, based on Grammatical Evolution (GE) and Koza's developmental expressions. GE is an evolutionary algorithm which can generate code in any programming language and uses variable length binary strings. The binary genome determines which production rules in a BNF grammar definition are used in a genotype to phenotype mapping process to a program. In fact, as far as the authors know, there is no work in the literature that applies this evolutionary paradigm to automatic design of analog circuits. A ten member subset of Koza's developmental functions has been also used, with a fixed transistor embryo. Evaluation of each circuit (individual) is done with NGSpice [7] which is based on Spice3 [6].

The rest of the paper is structured as follows: Section 2 describes our method in detail. Section 3 is devoted to the application of our method to a case study: the automatic synthesis of a BJT Common Emitter Amplifier, with a comparison to a human designed amplifier. Finally, conclusions and future work are given in Section 4.

2 Method Description

In this section, we present our proposal to the automatic design of electronic amplifiers. This includes a description of the grammar, the function set and the fitness function used. We also devote a short introduction to Grammatical Evolution, evolutionary paradigm in which our method is based.

2.1 Grammatical Evolution

Grammatical Evolution (GE) was introduced by Michael O'Neill and Conor Ryan [8], as an alternative to Genetic Programming. GE is an evolutionary algorithm which can generate code in any programming language and uses variable length binary strings, unlike Genetic Programming which (GP) is based on parse trees for representing genomes. Crossover and mutation operators work

over these binary strings and the decoding of each individual is based on a BNF grammar of the language. A BNF grammar is defined by a tuple of four elements: $\{S, T, N, R\}$ where S is the start symbol, T is the set of terminal symbols, N is the set of non-terminals and R is the set of rules of production.

GP needs special crossover and mutation operators to work over parse trees and to preserve their consistence as syntactically correct programs. This consideration is not needed in GE, which can use generic and standard operators, while the decoding process guarantees the correctness of the resulting program.

In GE, the unit of information is called codon, drawing inspiration from Biology. Normally, each byte is considered a codon, so it can have 256 possible values. The decoding of any string consists on traversing it, using each codon for each choice of the production rules of the language grammar. The chosen production rule is the one numbered with the codon modulus the number of choices that correspond to the rule that is being considered, as shown in equation 1. A decoding example is provided in subsection 2.2.

$$productionRule = codon \text{ MOD } numberOfRules \quad (1)$$

2.2 Koza's Developmental Functions

John R. Koza and his collaborators introduced a developmental process approach to represent and synthesize analog circuits using GP [4]. The process is based on a circuit embryo which is divided into a test fixture and the embryo itself. The test fixture is the part of the circuit which does not change during the developmental process and comprises, for example, the power supply, the signal source, the source resistor and the load resistor. The embryo itself comprises modifiable wires, which are modified by the developmental functions of the specific developmental expression. Developmental functions operate over a modifiable wire or component, transforming it into a new modifiable component or a mesh formed by new modifiable wires and/or components. Developmental expressions, made of developmental functions, are represented by GP parse trees, and codify the developmental process which leads to a final circuit.

Our approach combines GE with Koza's developmental expressions. First of all embryo and test fixture are shown, followed by our proposed developmental expressions grammar for decoding chromosomes into developmental expressions. A decoding example is provided and finally our proposed fitness function is shown.

Embryo and Test Fixture As will be shown in the next section, our goal is obtaining an one-stage common emitter amplifier, so we have modified the original embryo proposed by Koza, to contain one fixed transistor which cannot be modified by the developmental process. Only the connections to the transistor can be modified. The used embryo is set into a test fixture (fig. 1a) with a signal source, its associated source resistor, and a load resistor. It also includes a fixed transistor with four modifiable wires, Z_1 , Z_2 , Z_3 and Z_4 (fig. 1b).

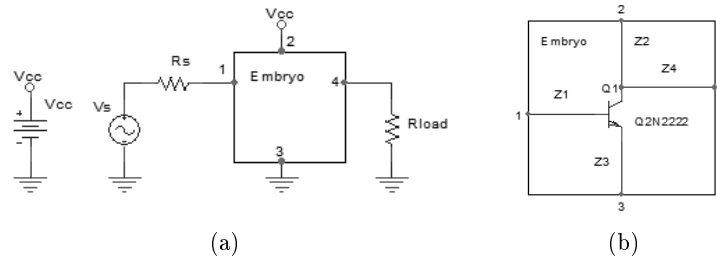


Fig. 1: Test fixture (a) and embryo (b)

Developmental Expressions Grammar Table 1 shows the grammar used for the developmental expressions in Extended Backus-Naur Form (EBNF). The start symbol is *devExpression* which produces a final developmental expression. The symbol *RPB* stands for result-producing branch which comprises the process of transformations for a determined modifiable wire or component.

S =	devExpression
T =	{'LIST','END', "CUT", "NOP", "PARALLEL", "SERIES", "THREE_GROUND", "PAIR_CONNECT", "THREE_VCC", "R", "C",'e', '0', '1', ..., '9', "-12", "-11", ..., "-3"}
N =	{RPB, FUN0, FUN1, FUN2, FUN3, CC, resistorValue, capacitorValue, digit, nonZeroDigit, capacitorExp}
R =	formed by the following rules of production
devExpression =	"LIST", '(', RPB, ')', { '(', RPB, ')' };
RPB =	FUN0 FUN1, '(', RPB, ')', FUN2, '(', RPB, ')', '(', RPB, ')', FUN3, '(', RPB, ')', '(', RPB, ')', '(', RPB, ')', '(', RPB, ')', CC, '(', RPB, ')';
FUN0 =	"END" "CUT";
FUN1 =	"NOP";
FUN2 =	"PARALLEL";
FUN3 =	"SERIES" "THREE_GROUND" "PAIR_CONNECT" "THREE_VCC";
CC =	("R", resistorValue, "C", capacitorValue);
resistorValue =	nonZeroDigit, '.', digit, 'e', digit;
capacitorValue =	nonZeroDigit, '.', digit, 'e', capacitorExp;
digit =	'0' '1' '2' '3' '4' '5' '6' '7' '8' '9';
nonZeroDigit =	'1' '2' '3' '4' '5' '6' '7' '8' '9';
capacitorExp =	"-12" "-11" "-10" "-9" "-8" "-7" "-6" "-5" "-4" "-3";

Table 1: Developmental expressions grammar

Decoding example The following example chromosome, [34, 5, 19, 30, 43, 15, 64, 68, 15, 15, 24, 25, 37, 45, 36, 44, 42, 81, 100, 104, 75, 8, 14, 12, 4, 17, 13, 90, 88], is decoded as expression: *LIST (C 2.0e-9 (END))(THREE_VCC(END))(CUT)*

(*END*))(*R 1.0e4 (END)*)(*R 5.7e3 (END)*). The first steps of the decoding process are as follows:

1. The decoding starts with *S* symbol, that is, with *devExpression*.
2. Since our embryo comprises four modifiable wires, our expression need to comprise four *RPB*'s, so production rule for *devExpression* expands as *LIST (RPB)(RPB)(RPB)(RPB)*
3. Production rule for first *RPB* has five choices and first codon value is 34, so $34 \text{ MOD } 5$ gives rule #4 which gives: *LIST (CC (RPB))(RPB)(RPB)(RPB)*
4. Expansion of rule for *CC* with codon value 5 ($5 \text{ MOD } 2$) gives rule #1 which is associated to *C* function, so the expression is now: *LIST (C capacitorValue (RPB))(RPB)(RPB)(RPB)*
5. Expansion of capacitorValue needs three codons: 19, 30 and 43, giving value $2.0e - 9$. *LIST (C 2.0e-9 (RPB))(RPB)(RPB)(RPB)*
6. Next is rule #0 for *RPB* which gives, *FUN0*. Now the expression is: *LIST (C 2.0e-9 (FUN0))(RPB)(RPB)(RPB)*
7. Next is rule #0 for *FUN0* which gives *END*. Now the expression is: *LIST (C 2.0e-9 (END))(RPB)(RPB)(RPB)*
8. Next is rule #0 for *RPB* which gives, *FUN0*. Now the expression is: *LIST (C capacitorValue (FUN0))(RPB)(RPB)(RPB)*
9. And so on.

Developmental function set As shown before, the grammar of our GE based implementation of Koza's developmental expressions uses a subset of only ten functions of all the functions defined by Koza. Table 2 shows the functions used.

Fitness function We have considered just five goal circuit parameters as specification for the amplifier: gain, low cut-off frequency, input and output impedances and dynamic margin. In order to reduce processing time, we have limited the measures of gain to three frequencies, and the impedances and dynamic margin are measured just to one frequency. Equation 2 shows the fitness function in this implementation. Where each term is the difference between the measure from the circuit and the design specification normalized by the latter value. So, the closer the measures are to the specified values, the closer *F* is to zero, making it a minimization problem.

$$F = \left[\sum_{i=1}^3 \Delta_{G_{dB}}(f_i) \right] + \Delta_{G_{dB}}(f_L) + \Delta_{Z_{in}}(f_{int1}) + \Delta_{Z_{out}}(f_{int1}) + \Delta_{DM}(f_{int2}) \quad (2)$$

The gain $G(f)$ is measured in decibels (dB) at three frequencies f_1 , f_2 and f_3 . The gain is also measured at the specified low cutoff frequency f_L . Input and output impedances, Z_{in} and Z_{out} are measured at an intermediate frequency, f_{int1} , and DM is measured at an intermediate frequency f_{int2} . Dynamic margin requires a transient analysis, which is measured with a sinusoidal input signal



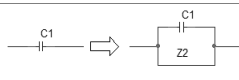
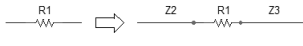

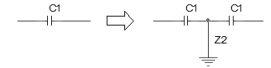

Developmental function	Arity	Description
END	0	End of a modifiable wire
CUT	0	Cut and ends a modifiable wire
NOP	1	No operation, the modifiable wire continues being modifiable
R	2	
C	2	
PARALLEL	2	
SERIES	3	
PAIR_CONNECT	3	
THREE_GROUND	3	
THREE_VCC	3	

Table 2: Developmental functions

having an amplitude high enough to saturate the output. We estimate the dynamic margin from the maximum and minimum saturated output values from one cycle, and we also use the output voltage when there is no input, from an operation point analysis. Equation 3 shows the expression for estimating the dynamic margin, where V_{max} is the maximum output voltage, V_{min} is the minimum output voltage, and V_c is the output voltage value when there is no input.

$$DM = 2 \min(V_{max} - V_c, V_c - V_{min}) \quad (3)$$

3 Case Study: BJT Common Emitter Amplifier

In this section, we present the design specifications of the target amplifier and the configuration parameters for the algorithm. Finally, we present the results of the experiments conducted and the best circuit obtained.

3.1 Design Specifications and Configuration Parameters

Table 3a shows the design specifications of the target amplifier and table 3b shows the GE-based algorithm parameters used. The values of the components V_{cc} , R_s and R_{load} are known in advance, that is, they form part of the test fixture (see fig. 1a) and are not considered in the evolutionary process. There is no

established fitness goal as a termination condition. Instead we allow running up the algorithm to the maximum number of generations. The algorithm parameters for the experiment conducted were obtained from preliminary runs. In these preliminary runs we also found that the algorithm was capable of giving good results without a duplication operator, which was introduced in GE [8]. So no duplication operator was used in the experiment.

<i>Gain</i>	f_L	Z_{in}	Z_{out}	DM	V_{cc}	R_s	R_{LOAD}
12dB	10Hz	20K Ω	6K Ω	18V	20V	600 Ω	100K Ω

(a)

Goal	Amplifier circuit
Embryo	Common emitter embryo
Representation	Variable length strings of codons (bytes)
Initialization	Random byte strings of 1-10 length
Length	1-10
Fitness function	see equation (2), where $f_1 = 1kHz$, $f_2 = 10kHz$, $f_3 = 100kHz$, $f_L = 10Hz$, and $f_{int1} = f_2$, $f_{int2} = f_1$
Grammar	see table 1
Function set	see table 2
Population	1000
Crossover	1 point, codon string, limited to 250 codons
Crossover probability	0.5
Mutation	Bitwise
Mutation probability	0.001
Parent selection	3 member-Tournament selection
Replacement selection	Generational
Elitism	2
Wrapping	4
Termination condition	250 generations
Execution number	51

(b)

Table 3: Design specifications (a) and algorithm parameters (b)

3.2 Results and Evaluation

Table 4a shows the algorithm performance measures including Success Rate (SR) and Mean Best Fitness (MBF), after 51 executions.

The criterion for success is fitness lower than 0.6. Higher values provide very low dynamic margin values, which are considered unacceptable. Performance results show a high MBF value and also a high standard deviation, nevertheless, since the problem is a design one, only one high quality solution is needed, so

this behavior is not an issue. In fact, an algorithm which provides few high quality solutions and bad solutions the rest of the time is preferred over an algorithm which provides just over average solutions every time. Table 4b shows the results of the three successful circuits, obtained by the evolutionary algorithm (first three rows). Also included the performance of a circuit designed by hand by an electronic engineer (fourth row). The results shown include measures of gain, drop at low cut-off frequency, input and output impedances and dynamic margin.

<i>Hits</i>	<i>SR</i>	<i>maxBF</i>	<i>minBF</i>	<i>MBF ± SD</i>	<i>Mean exec. time</i>
3	5.88%	45.880	0.438	7.402±14.102	6.3'

(a)

<i>Circuit</i>	<i>G_{dB}(f_i)</i>			<i>G_{dB}^{drop}(10Hz)</i>	<i>Z_{in}(10kHz)</i>	<i>Z_{out}(10kHz)</i>	<i>DM(1kHz)</i>	<i>Fitness</i>
	(1kHz)	(10kHz)	(100kHz)	(dB)	(kΩ)	(kΩ)	(V)	
<i>1</i>	12.02	12.00	10.60	-3.00	19.01	6.39	14.32	0.438
<i>2</i>	12.05	12.05	11.97	-3.01	9.36	5.92	17.98	0.559
<i>3</i>	11.94	11.93	11.08	-3.05	22.78	6.37	12.75	0.596
<i>human</i>	11.34	11.34	11.34	-1.55	25.29	5.99	14.79	0.698

(b)

```
LIST (THREE_VCC(END)(END)(C 3.2e-3 (C 2.4e-11 (PARALLEL(END)(C 3.5e-8 (C 1.8e-9 (R 7.5e5
(R 2.2e4 (END))))))))(R 6.4e7 (NOP(NOP(NOP(C 9.2e-9 (C 8.8e-11 (PARALLEL(END)(R 6.8e8
(NOP(C 5.8e-3 (C 7.3e-11 (PARALLEL(END)(R 6.4e3 (END))))))))))))(END)(C 1.5e-7 (END))
```

(c)

Table 4: Summary of results: Algorithm's performance global measures (a), results obtained by the three best evolutionary amplifiers and by a human designer (b), developmental expression of the best evolutionary circuit (c).

Circuit #1 is meets all of the goal criteria but the dynamic margin. Circuits #2 and #3 show a trade off between dynamic margin and input impedance.

Through an inspection of the parameters of the shown circuits, it seems that is somehow easy for the algorithm to provide good values for gain, low cut-off frequency and output impedance, while showing difficulties for meeting the goal criteria for dynamic margin and input impedance, showing a trade off between them in two cases. This behavior is also found in non successful circuits. Table 4c shows the best circuit developmental expression.

Human designed circuit shows an approximately met specifications, due to the design process itself in which some approximations have to be done, and the transistor, a non-linear component, has to be linearized by using an equivalent circuit. Calculated fitness value is slightly worse than the fitness value of the third circuit.

3.3 Best circuit analysis

Figure 2 shows the best evolutionary amplifier obtained (fig. 2a), and the amplifier obtained by the human designer (fig. 2b). No simplifications have been done to the best circuit. So, there are some components, like resistors in series or parallel, which could be simplified, or in some cases neglected due to their values. In this way, capacitors C_2 , C_3 and C_4 , due to their low capacity values (pF), only have an impact in high cut-off frequency (not considered in our analysis). They are probably allowed in the circuit due to lack of an explicit high cut-off frequency term in the fitness function, other than the gain criterion at $100kHz$. These capacitors could be neglected, while meeting all specifications criteria. It can also be seen that the circuit polarization network is poor, because this configuration is known to be very sensitive to variation in transistor's current gain. Better polarization networks are based on a voltage divider and also use an emitter resistor. We have also found that the used estimation of dynamic margin is misled by spurious peaks in transient analysis. Actual dynamic margin was measured to be $7V$ which is lower than the estimated. Another minor dysfunction is the absence of an input capacitor, so there is a bias current into the signal source.

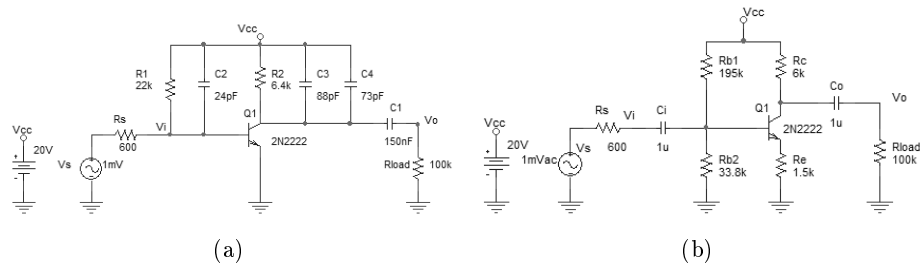


Fig. 2: Amplifiers obtained: best evolutionary circuit (a), human design (b)

4 Conclusions and Future Work

Grammatical evolution has been proved as a valid method for evolving one stage amplifiers using Koza's developmental expressions as the object language. A new grammar comprising ten developmental functions from Koza's set was used. The goal specification for the amplifier included five parameters: gain measured at three frequencies, low cut-off frequency, input and output impedances and dynamic margin. The embryo included a fixed transistor, in common emitter configuration, which cannot be modified during the developmental process. It also included four modifiable wires. The conducted experiment showed that the algorithm has a low success rate, but this is not an issue in a design problem,

because we can afford a high number of runs until getting a small set of good results. Three successful circuits have been obtained and their characteristics have been shown. The best circuit met all of the goal criteria but the dynamic margin. Comparison to a human designed amplifier shows promising results.

Future work could include a weighting of the fitness function terms because some deviations from the goal in one parameter may be more acceptable than in the others. The best obtained circuit showed a polarization network without feedback, which is known to be very sensitive to variation in transistor's current gain. Analysis of variation of this parameter should be introduced and taken into account in fitness function. A better estimation method is also needed for dynamic margin, since the used estimation is misled by spurious peaks.

Acknowledgement

This work was supported by funds of the Advanced Artificial Intelligence Master Program of the Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain.

References

1. A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, October 2008.
2. G.G.E. Gielen and R.A. Rutenbar. Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1854, dec 2000.
3. J.B. Grimbleby. Automatic analogue circuit synthesis using genetic algorithms. *Circuits, Devices and Systems, IEE Proceedings -*, 147(6):319–323, dec 2000.
4. J.R. Koza, D. Andre, F.H. Bennett, and M.A. Keane. *Genetic Programming III: Darwinian Invention & Problem Solving*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.
5. E. Martens and G. Gielen. Classification of analog synthesis tools based on their architecture selection mechanisms. *Integration, the VLSI Journal*, 41(2):238–252, 2008.
6. L. W. Nagel and D.O. Pederson. *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California, 1973.
7. P. Nenzi and H. Vogt. Ngspice users manual version 23, 2011.
8. M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5:349–358, 2001.
9. E. Tlelo-Cuautle and M. Duarte-Villaseñor. Evolutionary electronics: Automatic synthesis of analog circuits by genetic algorithms. In Ang Yang, Yin Shan, and Lam Bui, editors, *Success in Evolutionary Computation*, volume 92 of *Studies in Computational Intelligence*, pages 165–187. Springer Berlin / Heidelberg, 2008.
10. E. Tlelo-Cuautle, I. Guerra-Gómez, M. Duarte-Villaseñor, L.G. de La Fraga, G. Flores-Becerra, G. Reyes-Salgado, C.A. Reyes-García, and G. Rodríguez-Gómez. Applications of evolutionary algorithms in the design automation of analog integrated circuits. *Journal of Applied Sciences*, 10:1859–1872, December 2010.
11. R.S. Zebulum, M.A. Pacheco, and M. Vellasco. Comparison of different evolutionary methodologies applied to electronic filter design. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 434–439, may 1998.