Generation of new power processing structures exploiting genetic programming

G. Doménech-Asensi
Dpto. de Electronica y Tecnología de Computadoras
Universidad Politécnica de Cartagena
Cartagena, Spain
gines.domenech@upct.es

T. J. Kazmierski
Dpt. of Electronics and Computers Science
University of Southampton
Southampton, UK
tjk@ecs.soton.ac.uk

***Abstract*—This paper describes the use of genetic algorithms to generate power processing circuits. In order to speed up the algorithm, the fitness of the circuits is evaluated using an explicit integration method based on the 4th order Adams–Bashforth formula. Different combinations of genetic primitives for the crossover and mutation processes have been tested. The algorithm is demonstrated by generating new structures of voltage multipliers, which specifically focus on energy harvesting systems. These systems require low input voltages, usually under the diode threshold value. The Adams–Bashforth method allows to achieve a simulation time that is about five times faster than that of SPICE-based simulations.**

Keywords—evolutionary computation; power processing circuit;energy harvesting; analog simulation

I. INTRODUCTION

The field of evolutionary computation applied to analog circuit design has been the object of research for a long time. Among the targeted circuits, two fields have been mainly researched because of their widespread application. On the one hand, for high-frequency applications linear amplifiers have been widely analyzed [1][2], while on the other, for lower frequencies, analog passive filters synthesis has been researched. In [3] an automated passive filter genetic programming using a tree representation is proposed. This method assumes that the circuit can be analyzed using series and parallel transformations, which is not always possible because of the circuit topology. In [4] a framework for the synthesis of generic passive analog circuits is presented and demonstrated with two examples of passive filters. This method uses a SPICE-type netlist to describe the circuit and SPICE-based simulations to evaluate their fitness. In [5] an extended state of the art for analog filter synthesis based on genetic programming can be found.

This paper takes as starting point the work developed in [4] and aims to prove the capability of genetic programming to create new structures of power processing circuits. In this work the use of such circuits is focused in their application to energy harvester systems, which implies very low input voltages, below the diode voltage thresholds. In order to speed up the fitness evaluation, SPICE simulations are replaced by a explicit integration method based on the Adams–Bashforth formula. This technique lies on the linearization of the state equations of the analog components of the circuit and allows a bigger maximum step-size preserving the numerical stability of the integration algorithm [6]. The rest of the paper is organized as follows: Section II describes the genetic algorithm employed in this work. The algorithm is demonstrated with an example in Section III. Finally, conclusions are drawn in Section IV.

II. GENETIC ALGORITHM

Fig. 1.a shows the structure of the test circuit. The power processing circuit (PPC) created by the evolutionary algorithm is connected on one side to an input voltage source V_{in}, and on the other side to an output circuitry made of a series connected diode and capacitor. These two elements are, in fact, part of most of the existing PPCs. In this work, they have been placed as a common output circuitry for all the circuits under test, acting as a kind of load. The representation of the PPC is based on a SPICE-type netlist. Fig 1.b shows an example of circuit composed by a capacitor, C0, and a diode, D1 connected between nodes 1-3 and 3-0 respectively.

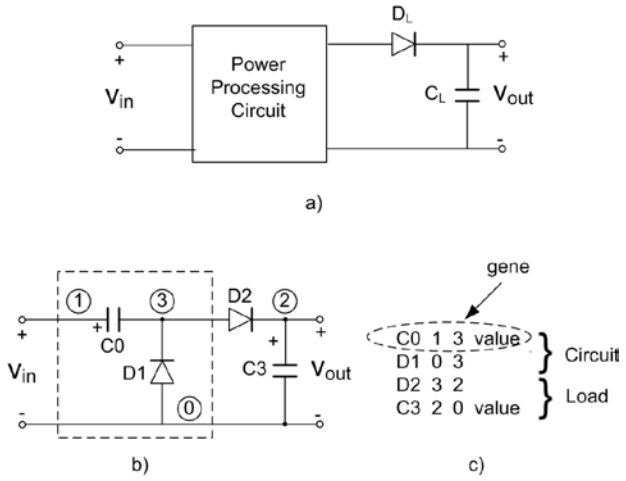

Fig. 1. Structure of the test circuit a) Test circuit b) Example of circuit with named devices and nodes c) Circuit representation.

This circuit is described in terms of a listing of devices (Fig 1.c) where D2 and C3 correspond to the output circuitry (Dl and CL) shown in Fig. 1.a. Regarding the genetic algorithm definitions, each line in the listing of Fig 1.c is a gene, while the set of all genes forms a chromosome. Each gene is composed of a type of device (C or D), a device identifier (0,1,…), two nodes, which identify the sense of the current through the component, and, in the case of capacitors, an additional field to specify its value. In this example, the chromosome consists of two genes (1st and 2nd lines), while lines 3rd and 4th correspond to the load circuit.

The structure of the genetic algorithm is described in Fig. 2. The algorithm comprises two main cascaded steps. In each step, the known processes of crossover, mutation, circuit fitness evaluation and selection of best candidates are performed. These steps end when a predefined number of generations has been created. The first step aims to generate evolved topologies from a set of parent circuits. During this step, device sizing is also performed.

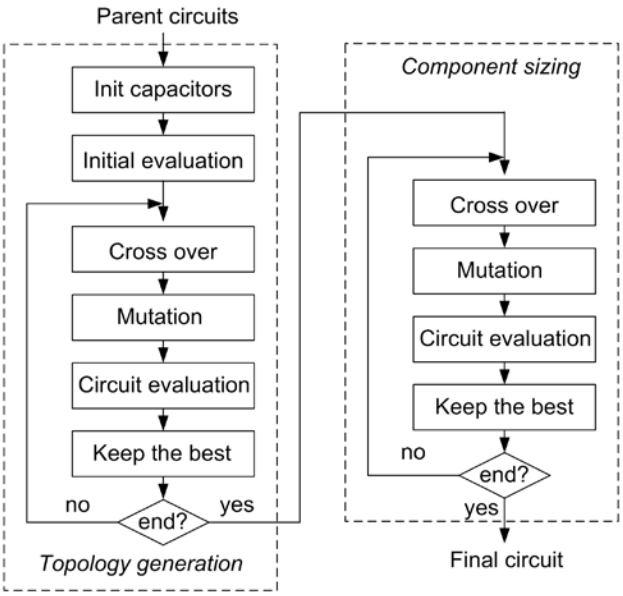

Fig. 2. Genetic algorithm flow.

TABLE I. LIST OF MUTATION PRIMITIVES

Primitive	Description
Mutate value	Changes the value of a component.
Switch C-D	Switches a diode by a capacitor or vice-versa.
Add serial node	Splits a randomly selected node into two nodes and creates a new node. A capacitor and a diode are respectively created between the new node and the two nodes resulting from the split one.
Add parallel device	Adds a device between two random nodes.
Remove device	Removes a device between two random nodes.

Once the topology with the best fitness is generated, it is replicated, including the capacitor values, to obtain a set of parent circuits, which then pass on to the second step of the genetic algorithm. This latter step will result in an accurately sized circuit.

A. Mutation and crossover

Mutation and crossover are the basis of genetic algorithms. In this work, mutation allows changes both in the topology of the circuit and in the values of the capacitors. Five types of mutation primitives have been used in the topology generation step. They are detailed in Table I. In the component sizing step, the only mutation primitive used is a modification of the capacitor value.

With regard to the crossover operation, during the topology generation step, it focuses on the circuit nodes. Fig. 3 shows an example of crossover between two parent circuits. Whenever two parents are randomly selected to perform a crossover, a random node is selected from each one. If these nodes have the same number of branches, then the crossover is carried out. If the number of branches is different, there are two more attempts to select random nodes from the parents before cancelling the crossover operation.

In the figure, nodes 3 and 4 have been randomly selected from parent 1 and 2 respectively. Since they have the same number of branches (three), the operation is performed. Devices D0, D1 and C2 from parent 1, in red, and devices D1, D2 and C0 form parent 2, in blue, are switched. They are then renumbered according to the parent device numbering. Now the two children circuits are ready.

The component sizing step comprises a crossover operation that consists of the exchange of capacitor values of randomly selected capacitors between two different parents.

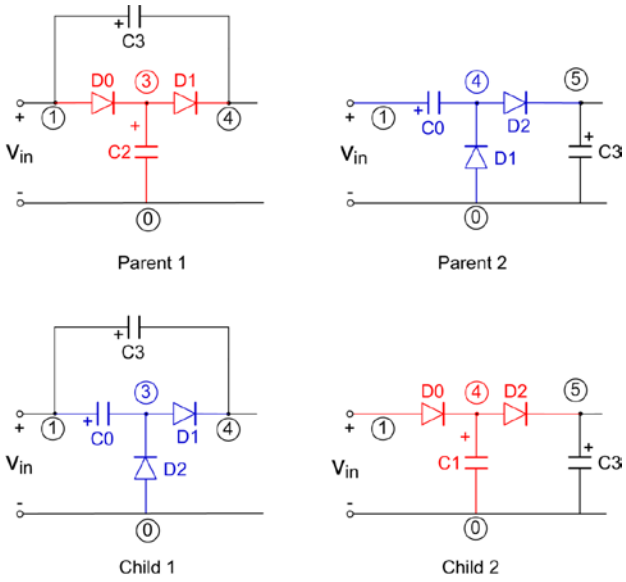

Fig. 3. Example of crossover between two parent circuits.

B. Circuit analysis

Simulations are performed employing a 4th order Adams–Bashforth integration scheme. This method is suitable for passive circuits and is faster than SPICE-type simulations [6].

Prior to simulating each circuit, it must be checked to verify that its topology is compatible with the integration method. This check comprises the following aspects: a) each node must be connected at least to one capacitor; b) there must be one and only one path from each node to either *Vin* or ground composed by nodes and capacitors; c) there must be no floating nodes; d) there must be no duplicated devices between two different nodes. In case a given circuit does not comply with these rules, it is subsequently modified. Additional capacitors are then added or removed, and isolated nodes are randomly renumbered using the number of an existing node. These modification rules provide additional mutation procedures to those already included in the genetic programming algorithm.

The integration method is based on the linearization of the state equations of the analog components of the circuit. Since power processing circuits comprise diodes, which are nonlinear components, it is necessary to linearize their models to create linearized state-space equations. Following the method proposed in [6], a linearized diode equation is created by differentiating the Boltzmann equation:

$$I_d = I_s\left(e^{Vd/Vt} - 1\right) \tag{1}$$

with reference to the diode voltage *Vd*. The linearized form of this equation is $Id = GVd + J$. In this form, $G = dI_d(V_d)/dV_d$, and $J = I_d(V_d)$ are the values of the equivalent conductance and ideal current source respectively, at the current time point. The values of *G* and *J* are previously calculated as piecewise linear functions of the voltage *Vd* and they are stored in look-up tables for different values of *Vd*. Following this technique, the models created are linear at each time point although *G* and *J* change.

The integration scheme requires five types of equations for each circuit. These equations are dynamically generated by the program and then evaluated using the integration scheme. The five types of equations are detailed taking the circuit of Fig 1 as an example. For this circuit, there are two state equations, one for each capacitor:

$$\begin{cases} dVc_0/dt &= Ic_0/C_0 \\ dVc_3/dt &= Ic_3/C_3 \end{cases} \tag{2}$$

Next, the characteristic functions of the diodes are created:

$$\begin{cases} Id_1 &= G_1Vd_1 + J_1 \\ Id_2 &= G_2Vd_2 + J_2 \end{cases} \tag{3}$$

The voltage drop in each diode is then computed, being *Vi* the voltage at the i[th] node:

$$\begin{cases} Vd_1 &= V_0 - V_3 \\ Vd_2 &= V_3 - V_2 \end{cases} \tag{4}$$

Then, the voltage at each node, except for nodes 0 and 1 (input), is computed. This voltage is calculated as the sum or the difference between an adjacent node voltage and the capacitor voltage placed in between:

$$\begin{cases} V_3 &= V_1 - Vc_0 \\ V_2 &= V_0 + Vc_3 \end{cases} \tag{5}$$

Finally, the current though each capacitor is evaluated. This current is computed as the sum of the currents at one of the nodes to which the capacitor is placed.

$$\begin{cases} Ic_0 &= -Id_1 + Id_2 \\ Ic_3 &= +Id_2 \end{cases} \tag{6}$$

Although these equations are dynamically evaluated, fact that could imply an increase of the simulation time, since no recompilation or scripts execution is needed, an overall saving of CPU time is obtained.

III. Example and Results

The method was tested for a set of 30 parent circuits and 500 generations both for topology evolution and for component accurate sizing. A low-frequency 50 Hz 250 mV sinusoidal voltage was applied to the circuit. This low-frequency voltage is common in electromechanical harvester systems, which are a primary application of this power processing circuit. Moreover, since it is desirable to obtain the highest output voltage, in each one of the two steps, the circuit fitness is evaluated performing a 10-second transient simulation. In order to prevent the effect of circuits with oscillating response due to certain time constants, the maximum average value of the output is used as fitness function instead of the output maximum instantaneous value. In this example, CL was 150 µF and the rest of capacitors could vary from 100 µF to 200 µF. The diode models used were all Schottky barrier diodes BAT85.

The probability of all the types of mutations was set to 15% after several tests. The probabilities of node crossover during the topology generation step and of value crossover during the sizing step were respectively set to 20% and 25%.

Fig. 4 shows the circuit with the best fitness after 500 generations. Altogether, the circuit and the load yield 5 capacitors and 5 diodes, the same number of devices as that of a conventional five-stage Dickson pump circuit. However, there are some differences between both circuits. C11 is parallel connected between nodes 3 and 5, instead of being connected between nodes 5 and 0, and C8 is connected to node 4 instead of node 1. Table II shows the values of the four capacitors after the topology generation step and after the component sizing step. It can be noted that the value of C4 tended to the value of CL while the other three capacitors evolved to the lower possible values (100 µF).

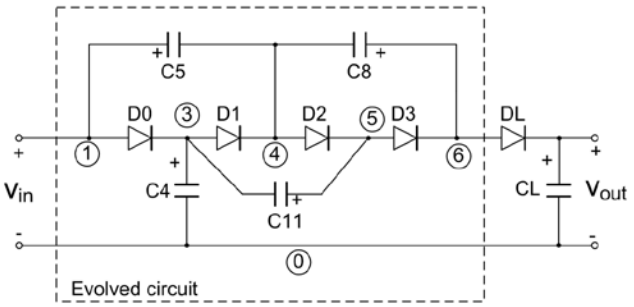

Fig. 4. Circuit with best fitness after 500 generations.

TABLE II. CAPACITOR VALUES

Capacitor	After Topology generation	After Component Sizing
C4	118.122 µF	147.37 µF
C5	138.45 µF	100.659 µF
C8	101.462 µF	101.462 µF
C11	15 µF	100.02 µF

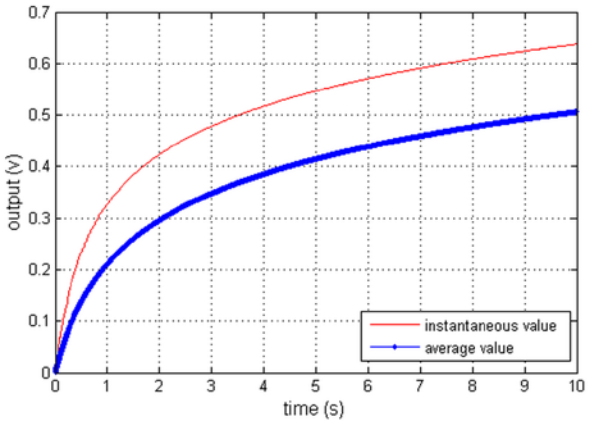

Fig. 5. Instantaneous and average output voltage of circuit with best fitness.

Fig. 5 displays the output and average voltages of the generated circuit. This circuit reaches a monotonically growth of up to 635 mV for 10 seconds. The best fitness, measured over the average voltage value is 506.15 mV. For a simulation step of 10 µs, the simulation time using the Adams–Bashforth for a single circuit was 3.76 s, almost five times faster than in the case of PSPICE (17.27 s).

IV. CONCLUSION

In this paper, the capability of genetic algorithms to generate novel voltage multiplier structures has been proven. During the development of the testbench, different genetic primitives were tested for mutation and crossover operations, however, this work only describes those which have proven to be more efficient. A mutation primitive focused on the creation of parallel nodes (i.e. new branches from a randomly selected existing node to ground, with a series connected capacitor and diode) was discarded because circuits tended to have a large number of devices in exchange of minimum performance improvements. Moreover, after several tests, the two-stage structure of the algorithm, with a first step focused on topology generation and a second step aimed at obtaining accurate capacitor sizing, proved to yield better results than the single-step procedure combining topology generation and device sizing.

The voltage multiplier circuit obtained with the shown example has the same number of devices as the classical five-stage Dickson pump but yields a higher voltage. In order to improve the performance of this algorithm, alternative genetic primitives for mutation and crossover operations can be developed and tested. Likewise, the fitness function can also be modified in order to measure other circuit performances. In this work, the fitness function evaluated the average output voltage in order to avoid the effect of circuits having oscillating behavior. An alternative function was also tested, weighing this value with the number of devices, in order to reward smaller circuits. However, the algorithm tended to generate very small circuits with poor output voltages. A space state exploration focused on balancing these two performances could yield new results.

ACKNOWLEDGMENT

This work was partially funded by Spanish government project TEC2015-66878-C3-2-R (MINECO/FEDER, UE) and by a research stay through the Campus de Excelencia Internacional Mare Nostrum

REFERENCES

[1] B. Liu, N. Deferm, D. Zhao, P. Reynaert and G. G. E. Gielen, "An Efficient High-Frequency Linear RF Amplifier Synthesis Method Based on Evolutionary Computation and Machine Learning Techniques," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 981-993, July 2012.

[2] B. Liu, D. Zhao, P. Reynaert and G. G. E. Gielen, "Synthesis of Integrated Passive Components for High-Frequency RF ICs Based on Evolutionary Computation and Machine Learning Techniques," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 10, pp. 1458-1468, Oct. 2011.

[3] S.J. Chang, H.-S. Hou and Y.K. Su, "Automated passive filter synthesis using a novel tree representation and genetic programming," IEEE Trans. on Evolutionary Computation, vol. 10, no. 1, pp. 93-100, Feb. 2006.

[4] A. Das and R. Vemuri, "An Automated Passive Analog Circuit Synthesis Framework using Genetic Algorithms," IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07), Porto Alegre, 2007, pp. 145-152.

[5] R.A. Vural, T. Yildirim, T. Kadioglu, A. Basargan, "Performance Evaluation of Evolutionary Algorithms for Optimal Filter Design", IEEE Trans. On Evolutionary Computation, vol. 16, no. 1, pp 135-147, Feb. 2012

[6] T. J. Kazmierski, L. Wang, B. M. Al-Hashimi and G. V. Merrett, "An Explicit Linearized State-Space Technique for Accelerated Simulation of Electromagnetic Vibration Energy Harvesters," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 4, pp. 522-531, April 2012.