

ANALOG CIRCUIT DESIGN AUTOMATION USING NEURAL NETWORK-BASED TWO-LEVEL GENETIC PROGRAMMING

FENG WANG^{1,2}, YUAN-XIANG LI²

¹Department of Computer Science, Wuhan University, Wuhan, 430072, China

²State Key Lab of Software Engineering, Wuhan University, Wuhan, 430072, China

E-MAIL: wangfengwhu@hotmail.com, yxli@whu.edu.cn

Abstract:

The design of analog circuits starts with a high-level statement of the circuit's desired behavior and requires creating a circuit that satisfies the specified design goals. The difficulty of the problem of analog circuit design is well known, and there is no previously known general automated technique to design an analog circuit from a high-level statement of the circuit's desired behavior. This paper proposes a two-layer evolutionary scheme based on Genetic Programming (GP) and Neural Network (NN), which uses a divide-and-conquer approach to design the analog circuits. Corresponding to the NN-TLGP, a new representation of circuit has been proposed here and it is more helpful to generate expectant circuit graphs. This algorithm can perform the circuits with dynamical size, circuit topology, and component values. The experimental results on the two design work show that this algorithm is efficient.

Keywords:

Evolutionary computation; two-level genetic programming; evolvable hardware; neural network

1. Introduction

Analog circuit design (ACD) plays an important role in electronic systems. The techniques for automating analog circuit design appeared about twenty years ago. By far, some techniques from intelligent computation, such as the (NN) have been under studying, and these techniques are indeed helpful to solve some complex problems [1,2]. Due to the complexity of the problem, it is very difficult to design the NN by the conventional methods. While the distribution of the processing cells of networks is more complex, it becomes more unfeasible to certify the main parameters of the NN, including the number of layers, the number of the cells of each layer and the interactions among the layers.

Meanwhile, much progression has been made in automating analog circuit synthesis using optimization algorithms [3,4]. Some researchers have used the

evolutionary computation, such as the genetic algorithms (GAs), to automate the NN design of circuits [5]. However, if the evolving size of the circuit is larger, the evaluation time is longer consequently. It will affect the efficiency of the generation of the good circuits. When a neural network uses a genetic algorithm for training, there is an increase in computational time, but compared with simple gradient descent, optimization does not fall into local minimum and be more accurate in prediction.

In this paper, we propose a NN-TLGP scheme to design the analog circuits. This scheme which used a divide-and-conquer approach is a marriage of Genetic Programming and Neural Network. Section two concentrates on how the hybrid scheme is applied to generate common analog circuits, especially on how to select component values and topology sizes for a given circuit topology. Section three describes the operations of the design work. The fitness function is given in section four. Section five describes two specific applications of the NN-TLGP scheme with the results and the analyses of them. Conclusions and plans for the future work are given in Section six.

2. Neural Network-Based Two-level Genetic Programming (NN-TLGP)

2.1. Basic Ideas of NN-TLGP

Based on the common GP, we employ a divide-and-conquer approach [6] to make the circuit divided into two levels before evolving, and name this new method NN-TLGP. The first level is neural network level, and the second level is the neuron level. In this way, we can make sure that, while the size of the circuit and its population become larger, the actual evolving size of the circuit's representation and the evolutionary computation will increase to a lower extent compared to the original method.

2.1.1. Neural Network Level

At the beginning of the evolution, we divide the circuit into different parts according to the sub-functions of the circuits. After the process of defining the structures of each neuron to be evolved in the neuron layer, these small neurons can be connected by some links to form a graph as the original circuit. All the links in a neural network can be seen as a linking method at one time. Both the separate physical component and the small sub-circuit neuron can be described as a node which represents a logic function set. These nodes are connected by the linking methods. Those different linking methods can be trained by the corresponding training algorithm to form a higher level topology, and we call each topology a neural network. Once a fully connected neural network is trained, its weight can be obtained. That's to say, if the neurons' structure is defined, we can get a more suitable circuit (evaluating by its fitness) by training the linking methods.

2.1.2. Neuron Level

As we mentioned above, in the circuit design, we divide the circuit into different parts according to the sub-functions of the circuits, each sub-function can be seen as a neuron in a neural network. After encoding each neuron into a genetic tree, we create more trees to make every neuron into a small genetic tree population by the three main operators of GP in the neurons during the circuit's evolution. Each individual as a genetic tree stands for a smaller specific circuit in the population of genetic trees. Each neuron is evolved according to its own function and has only one function tree called the main tree. That means the only output of one neuron is the root node of the main tree. Neurons are connected by making the output(s) of one neuron be the input(s) of other neuron(s). In an analog circuit evolving task, each node can be replaced by R (resistor), C (capacitor), L (inductor), or transistor configuration, which can be evolved by the three main operators of GP during the neuron's evolution.

As the NN-TLGP scheme mentioned above, an overview of the analog circuit design process is depicted in Figure 1.

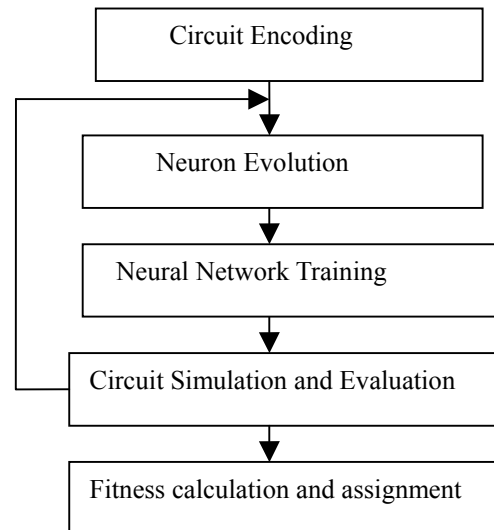


Figure 1. Overview of circuit design process

2.2. Circuit Representation and Encoding

One of the most important aspects of Genetic Programming is the strategy of encoding a solution. The encoding methods can directly affect the ability of the iterative process to converge on an appropriate solution. For a n -component circuit, a reasonable upper bound would be $O(n)$. Because of the direct relationship between the evolvable hardware's population size and the chromosomes' length, it is very important to shorten the length of the chromosomes. We also designed an encoding scheme base on our NN-TLGP. The encoding scheme can be described as two parts, one is the neural network encoding which aims at the structure of the circuit. The other is the neuron encoding which aims at the parameters of the circuit. The scheme can easily reduce the evolvable circuit's size to a much smaller one than it was before.

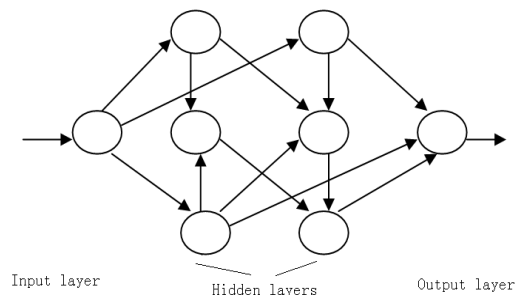


Figure 2. Neural network structure representation

Figure 2 depicts the structure of a neural network in our NN-TLGP, each node which represents a neuron was

connected to form a whole circuit. The nodes are connected by the linking methods, which would be trained during the neural network training process. Each node can be described in a netlist file as: [NodeName: linking method1, value1; linking method2, value 2; ...; linking method n, value n].

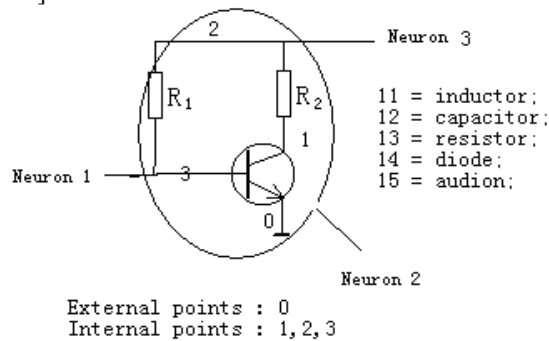


Figure 3. Neuron representation

Figure 3 depicts an example of the common emitter amplifier circuit. In this example, the chromosome consists of two genes. The genes determine the type, values of the physical characters of the component and links between the related components. The links are connected by the connecting points which are also the inputs or outputs of modules. Each connecting point may be classified as internal or external. While the former does not serve for any special purpose, the latter is connected to one of the following signals: power supply, ground, input signal or probed output. Supposing we divide the whole circuit into several parts, each of which is a neuron. The graph we have shown above is a part of one circuit, which is called neuron 2, and it connects to neuron 1, neuron 3 and ground. So the resistor R1 and R2 can be represented separately as [(2, 3), (13, 10Ω)] and [(1, 2), (13, 10Ω)], while the transistor can be represented as [(0, 1, 2), (15, -)].

3. Operations of NN-TLGP

3.1. Neural Network Training

Neural networks have an advantage over conventional technologies because they can solve complex problems that do not have algorithmic solution or for which an algorithmic solution is too complex to be found. NNs are trained by example instead of rules and are automated. When used in the analog circuit design, they are capable of rapid identification, analysis in real time.

The most widely used architecture of an NN is that of a multilayer perceptron (MLP) trained using Back

Propagation (BP) algorithm [7]. It is a gradient descent algorithm that tries to maximize the weights of the linking methods of the network. As Figure 2 shows, perceptron neurons grouped in several layers constitute the network structure. The first layer is the input layer, in charge of providing the external activations to the network (for example, the power supply). Placed on the extremely right side of the network is the output layer, which returns the results obtained by processing the input activations. Between those layers are several hidden layers, composed of a variable number of neurons. Each neuron in a certain layer receives its input activations from the neurons in the preceding layer. Each gives its output to the units in the next layer.

In the neural network level, the neural network is trained to optimize the weights of the linking methods by using the parallel perturbative weight update rule. Here, the linking methods, which are obtained from the circuit topology are now normalized and fed as inputs of the neurons in the neural network. At first, in the circuit initialization, the weights of the linking methods are generated randomly with the probability distribution: $\exp(-|r|)$, $r \in (0,1)$. This algorithm is based on some experimental results. And after the convergences of the NN, we can see that the weights of the linking methods are normally small, and a part of them are relatively big. It can make sure that this algorithm can search all the feasible solutions of the circuit, and enable the optimization does not fall into local minimum. Then, after the initialization, all input training weights are applied and the error is accumulated. This error is then checked to see if it was higher or lower than the unperturbed iteration. If the error is lower, the perturbations are kept, otherwise they are discarded. This process repeats until a sufficiently low error is achieved.

3.2. Neuron Evolution

In practice, the embryonic circuit is needed to be given before the operation. And the embryonic circuits should include some basic characters of the circuit by some transcendental knowledge. Such as the number of main functions of the circuit, some physical characters and model parameters about basic analog components and their initial evolution probabilities. The evolution probabilities of the model parameters might be changed during the following evolution process due to two main aspects: one is the importance of the correlative parameters; the other is the difficulties about techniques in the execution of the circuit components.

In the neuron level, the component state and the

neuron's structure can be changed by the following six operators: mutation, exchange, crossover, creation, upgrade, pruning.

- mutation : the mutation can be implemented by changing the physical parameters or the type of the component. For example, the resistor can be changed to a capacitor, but the nodes beside them don't have to change.
- exchange : swap the sub-trees under the selected nodes to generate two offspring belonging to the new population.
- crossover : first select two trees from the whole trees inside the module , then randomly select one node from each of these trees, do the same operation as the exchange operator.
- create : randomly generate a tree according to the appearance probability of the nodes in various layers.
- upgrade : select a tree whose fitness is the best in the module , get a sub-tree out of it as a new tree.
- prune : select a tree whose fitness is the worst in the module , delete one branch of the tree , then put the new tree into the genetic tree population inside the module.

4. Fitness Function

A fitness function must be devised for each problem before evaluation. Here we use a fitness function as,

$$fitness = \max f(x) = f(g, s, t) \quad (1)$$

$$f(g, s, t) = g^{c1} s^{-c2} t^{-c3} \quad (2)$$

where $c1$, $c2$ and $c3$ are all nonnegative real numbers. The sub-object g represents the performance of the evolving circuit, the sub-object s represents the complexity of the circuit (which can be also described as the circuit size), and the sub-object t represents the running time of the circuit [8].

While the fitness function can be described as three sub-objects g , s and t , these three sub-objects also have respective meanings. The sub-object g , which represents the performance of the evolving circuit, can be defined as,

$$g = \sum_{i=1}^N \alpha_i g_i, \quad (3)$$

N is number of the sub-objects while evaluating the performance of the circuit, which can also be defined by the concrete circuit. α_i can be adjusted adaptively during the evolutionary process, and apparently they satisfy,

$$\sum_{i=1}^N \alpha_i = 1 \quad (4)$$

We give each α_i a normal value due to the corresponding g_i at the beginning of the neuron circuit's evolution, and α_i can adjust during the following evolution process.

The sub-object s , which represents the complexity of the circuit, can be defined as,

$$s = p + q \quad (5)$$

$$p = \frac{n_{aud} + n_{amp}}{n_{total}}, \quad q = \frac{n_{feedback}}{N_{max}} \quad (6)$$

where n_{aud} is the number of the audions, n_{amp} is the number of the operational amplifiers, n_{total} is the total number of elements in the circuit, $n_{feedback}$ is the number of the feedback of the circuit, and N_{max} is the anticipative maximum number of the feedback of the circuit.

The sub-object t , which represents the running time of the circuit, can also be defined as,

$$t = \frac{T_{total}}{gen} \quad (7)$$

where T_{total} is the total time to get the result of the circuit, including the time for evolving and training and the time for simulating. gen is the current evolution generation of the neuron circuit.

In the evaluation of the analog circuit, we use the public-domain Berkeley SPICE (Simulation Program with Integrated Circuit Emphasis) circuit simulation program to simulate our circuits [9].

5. Experimental Results

This section is about the application of NN-TLGP for the solution of several practical analog circuit applications. At the beginning of the each evolution of the neuron, a basic embryo circuit is given as the primitive individual of the circuit. The embryo circuit is given due to the original function of the circuit, which is encoded by our two level encoding scheme.

In the neural network level, we used a particular multilayer perceptron structure. There are two degrees of freedom for determining it. These are the network size (number of layers and number of neurons) and the weights of the linking methods of the neurons, which convert the neuron's input activations. For network size, there is no deterministic way to find the most suitable structure for solving a particular problem. In general, the complexity of the circuit and desired circuit performance will mainly determine the network size. In the neuron level, the number of neurons will be changed during the evolving process. Whether it increases or decreases, the number is determined

by the fitness of each neuron which evaluated by the fitness function.

As mentioned above, the fitness function involved here is $f(g, s, t)$, s and t can be synchronously obtained by simulation. And some parameters can be initialized as follows: $P_{m0}=0.7$, $P_{c0}=0.1$, $k_1 = k_2=2$.

We chose the voltage amplifier and the low-pass filter, which are the most typical analog circuits, as the experimental circuits. In our design task, the network size was determined by the actual circuit, and the population size of the neuron level was set 1000. An initial circuit described by the netlist file using the two level encoding scheme was given before the formal design.

In the amplifier design experiment, we present the highest performance circuit found across the runs. The goal was to design an inverting amplifier capable of a dc voltage gain up to a maximum of 350 dB, while minimizing dc bias and maximizing linearity over the dc gain. The maximum gain was set to by using the feedback resistors. An error value is computed as the sum of the dc gain penalty (the target gain minus the observed gain), the dc bias (zero dc bias is ideal), and the degree to which the dc gain is linear. Figure 4 shows the schematic for the amplifier which has the highest fitness. Figure 5 shows the frequency response of the amplifier.

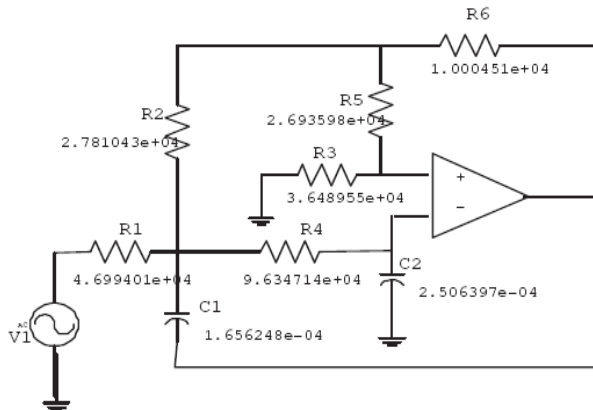


Figure 4. The schematic for the highest fitness evolved amplifier

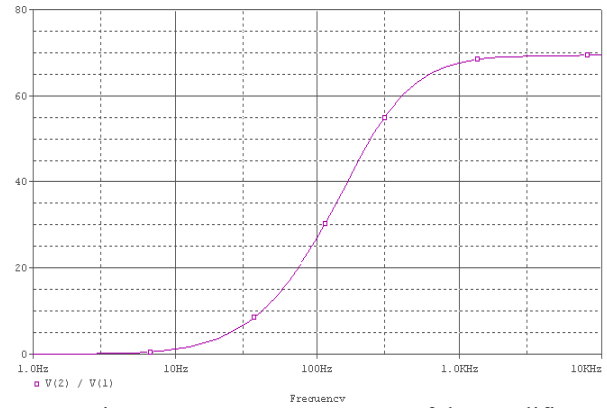


Figure 5. Frequency response of the amplifier

For the filter design experiment, the task had the specifications that satisfy the voltage gain is 20 dB and transition frequency is 30 kHz. Fitness was calculated to promote the regression of the evolved circuit's frequency response toward that of the target. Error values were computed as the absolute value of the difference of the individual's output. These error values were summed across evaluation points and act as a part of the fitness function. The evolved circuit is shown in Figure 6 and its frequency response is seen in Figure 7.

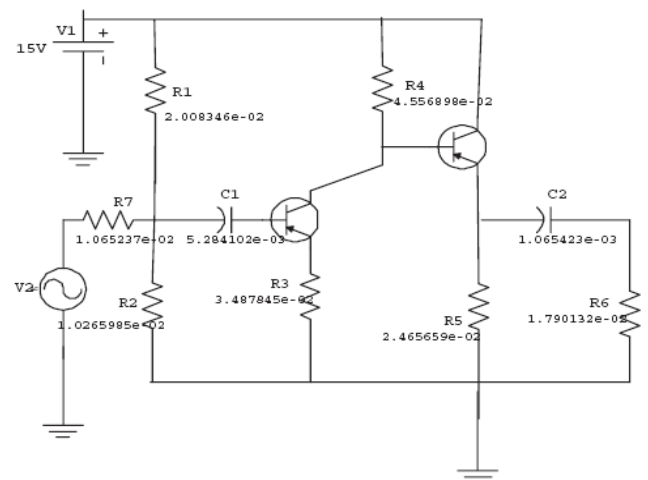


Figure 6. The schematic for the highest fitness evolved low-pass filter

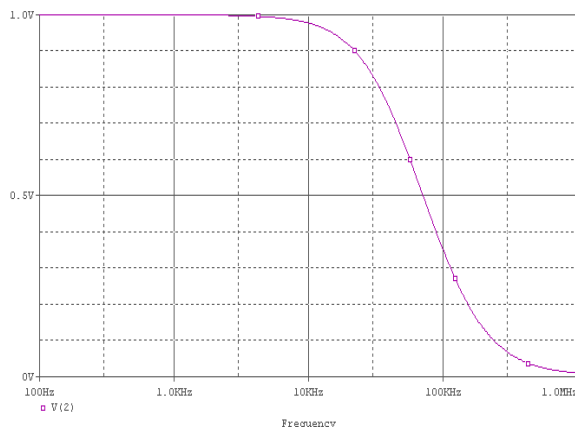


Figure 7. Frequency response of the low-pass filter.

In general, both the amplifier and low-pass filter can be generated quickly. In our evolution environment, whose PIV is 2.3 G, EMS memory is 1G Bytes and WINXP operating system, getting the best amplifier circuit merely cost 36 minutes, as well as 41 minutes for the low-pass filter, which is much faster than the technique that merely uses GA [10].

6. Conclusions

In this paper, we propose a scheme based on Neural Network and Genetic Programming (NN-TLGP), which uses a divide-and-conquer approach to design the analog circuit. We have shown that the corresponding circuit representation, operation strategy and adaptive fitness function based on the NN-TLGP scheme can automatically produce circuits in two applications. Detailed simulations of the designs suggest that the performances of the circuits are electrically well-behaved and thus are suitable for physical implementation. The experimental results show that using this algorithm can automate the analog circuit design, and it is faster than that just uses genetic algorithms.

Acknowledgements

This work described in this paper was supported by the National Natural Science Key Foundation of China with the Grant No.60133010, the National Research Foundation for the Doctoral Program of Higher Education of China with the Grant No.2003048604.

References

- [1] Koosh V.F., Goodman R.M., "Analog VLSI Neural Network with Digital Signal Processing", IEEE Transactions on Circuits and Systems II, Vol 49, No. 5, pp. 359-368, 2002.
- [2] Erten G., Goodman R.M., "Analog VLSI Implementation for Stereo Correspondence Between 2-D Images", IEEE Transactions on Neural Networks, Vol 7, No. 2, pp. 266-277, 1996.
- [3] Gielen G., Sansen W., "Symbolic Analysis for Automated Design of Analog Integrated Circuits", Boston, MA: Kluwer, 1991.
- [4] Sussman G.J., Stallman R.M., "Heuristic Techniques in Computer-Aided Circuit Analysis", IEEE Transactions on Circuits and Systems, Vol 22, 1975.
- [5] Koosh V., Goodman R., "VLSI Neural Network with Digital Weights and Analog Multipliers", Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS-2001), Sydney, Australia, Vol 2, pp. 233-236, 2001.
- [6] Torresen J., "A Divide-and-Conquer Approach to Evolvable Hardware", in 2nd International Conference on Evolvable Systems (ICES98), Lecture Notes in Computer Science, pp. 57-65, 1998.
- [7] Alspector J., Meir R., Yuhas B., Jayakumar A., "A Parallel Gradient Descent Method for Learning in Analog VLSI Neural Networks. Advances in Neural Information Processing Systems", San Mateo, CA: Morgan Kaufman Publishers, Vol 5, pp. 836-844, 1993.
- [8] Wang F., Li Y., "Multi-objective Adaptive Scheme for Analog Circuit Design Based on Two-layer Genetic Programming", International Conference on Neural Networks and Brain (ICNN&B '05), Vol 1, pp.274 – 278, 2005.
- [9] Quarles T., Newton A.R., Pederson D.O., A Sangiovanni-Vincentelli, SPICE 3 Version 3F5 User's Manual, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1994.
- [10] Lohn J.D., Colombano S.P., "A Circuit Representation Technique for Automated Circuit Design", IEEE Transactions on Evolutionary Computation, Vol 3, No. 3, pp. 205 –219, 1999.