Paper TEC #26 – Version 4 - Submitted June 12, 1997 to *IEEE Transactions on Evolutionary Computation*.

Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming

John R. Koza, *Member*, *IEEE*, Forrest H Bennett III, *Member*, *IEEE*, David Andre, Martin A. Keane, *Member*, *IEEE*, Frank Dunlap, *Member*, *IEEE*

Abstract – The design (synthesis) of analog electrical circuits starts with a high-level statement of the circuit's desired behavior and requires creating a circuit that satisfies the specified design goals. Analog circuit synthesis entails the creation of both the topology and the sizing (numerical values) of all of the circuit's components. The difficulty of the problem of analog circuit synthesis is well known and there is no previously known general automated technique for synthesizing an analog circuit from a high-level statement of the circuit's desired behavior.

This paper presents a single uniform approach using genetic programming for the automatic synthesis of both the topology and sizing of a suite of eight different prototypical analog circuits, including a lowpass filter, a crossover (woofer and tweeter) filter, a source identification circuit, an amplifier, a computational circuit, a time-optimal controller circuit, a temperature-sensing circuit, and a voltage reference circuit.

The problem-specific information required for each of the eight problems is minimal and consists primarily of the number of inputs and outputs of the desired circuit, the types of available components, and a fitness measure that restates the high-level statement of the circuit's desired behavior as a measurable mathematical quantity.

The eight genetically evolved circuits constitute an instance of an evolutionary computation technique producing results on a task that is usually thought of as requiring human intelligence. The fact that a single uniform approach yielded a satisfactory design for each of the eight circuits as well as the fact that a satisfactory design was created on the first or second run of each problem are evidence for the general applicability of genetic programming for solving the problem of automatic synthesis of analog electrical circuits.

Index Terms – Design automation, genetic programming, analog circuit synthesis, electrical circuits.

J. Koza is with the Computer Science Department, Stanford University, Stanford, California 94305 (e-mail: koza@cs.stanford.edu; URL: http://www-cs-faculty.stanford.edu/~koza/).

Forrest H Bennett III is a visiting scholar with Computer Science Department, Stanford University, Stanford, California 94305 (e-mail: forrest@evolute.com).

David Andre is with Computer Science Division, University of California, Berkeley, California (e-mail: dandre@cs.berkeley.edu)

Martin A. Keane is with Martin Keane Inc., 5733 West Grover, Chicago, Illinois 60630 (e-mail: makeane@ix.netcom.com).

Frank Dunlap is with Enabling Technology, Inc., Palo Alto, California (e-mail: Frank_Dunlap@compuserve.com).

1. INTRODUCTION

Automatic programming is one of the central goals of computer science. Samuel [1] described this goal as making computers perform required tasks without being told explicitly how to accomplish these tasks. A particularly challenging area for applying automatic programming involves the design (i.e., synthesis) of analog electrical circuits. Typically, design necessitates creation of a complex structure to satisfy user-defined requirements. The requirements indicate what needs to be done, while the design details how to do it. Circuit design is a significant activity of practicing electrical engineers and is commonly viewed as requiring human intelligence.

The design process for analog circuits begins with a high-level description of the circuit's desired behavior. Both the topology and the sizing of the circuit must be chosen such that the resulting circuit satisfies the design objectives. The topology comprises the gross number of components in the circuit, the type of each component (e.g., a resistor), and a list of all connections between components. The sizing entails specifying the values (often numerical) of each component in the circuit.

Circuit design is of considerable practical importance because all electrical circuits are ultimately analog circuits. Moreover, a layer of analog circuitry provides an interface between most digital circuits and the rest of the world. Further, all commonly used analog circuits must be redesigned with the introduction of each new generation of solid-state process technology.

Progress has been made in automating the design of certain categories of purely digital circuits; however, the design of analog circuits and mixed analog-digital circuits has not proved as amenable to automation [2]. When describing "the analog dilemma," Aaserud and Nielsen [3] noted "Analog designs are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts and so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product. ... Analog circuit design is ... a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a from of art rather than science." As a result there is a shortage of design engineers.

There has been no general automated technique for synthesizing an analog electrical circuit from a high-level statement of the desired behavior of the circuit. The difficulty of this problem is well known. In fact, even the analysis of the behavior of an already designed circuit is considerably more difficult for analog circuits than for digital circuits. The behavior of an analog circuit is specified by a system of integro-differential equations (IDEs) (one equation for each node or loop in the circuit in accordance with Kirchhoff's laws). When the circuit contains only linear and inactive components, such as capacitors, inductors, and resistors, the equations in the system are merely linear IDEs and mathematical techniques (e.g., Laplace transforms) may, in principle, be used to solve the equations. But when the circuit contains nonlinear and active components (e.g., transistors or diodes), the equations become nonlinear. In practice, the behavior of analog circuits is analyzed using numerical simulation. The problem of designing (i.e., synthesizing) a circuit with a particular desired behavior is even more vexatious than the problem of analyzing the behavior of an existing circuit because synthesis calls for creating both the topology and the sizing of each component in the circuit.

When engineers design complex circuits, they bring human knowledge, intelligence, experience, and intuition to bear on the problem. The engineers' approach is very different from the process used in nature to design the most complex known structures: living organisms. In

nature, the design process proceeds by evolution — random variation and natural selection. It does not follow the organized, thought-out approach of the engineer. The question posed in the current work is whether the analog of nature's design process, evolutionary computation, can be successfully applied to the problem of analog circuit design.

This paper presents a single, uniform approach to the automatic synthesis of both the topology and sizing of analog electrical circuits using genetic programming. The approach is illustrated by applying it to eight types of circuits. Section 2 provides background on synthesizing circuits. Section 3 presents the eight test problems. Section 4 describes the method for developing a circuit starting from an embryonic circuit using genetic programming. Section 5 details preparatory steps that are required prior to evolution. Section 6 offers results on the eight problems, emphasizing the evolution of a voltage reference circuit. Detailed depiction of evolved circuits are offered for readers familiar with circuit design. Section 7 summarizes, giving particular attention to the uniformity of approach and its general applicability to the automatic synthesis of analog electrical circuits.

2. PREVIOUS WORK

Numerous techniques have been applied to various parts of the problem of circuit synthesis. An early effort concerned the use of constraint propagation [4], but in the 1980s attention was focused on knowledge-based design of operational amplifiers and comparators based on user-supplied circuit topologies and associated design knowledge [5,6] (also see [7,8] for an extended knowledge-based system that attempted to find an appropriate sizing given a topology and could create new topologies when deemed necessary). Recent efforts in [9,10] start with the circuit topology and performance specifications. The system first produces an executable performance prediction module that is customized for the particular problem at hand. The prediction module guides simulated annealing for the sizing and biasing of a satisfactory circuit. This system was enhanced [11] to be capable of modifying a user-supplied topology while sizing and biasing. Binary variables indicate the presence or absence of certain connections and components in the topology. Again, annealing is used to perturb the topology by removing elements from a user-supplied superset.

The possibility of applying evolutionary computation to design problems was recognized in the earliest pioneering work, including efforts in the 1960s [12, 13]. These methods have been extended to evolve the length and structure of recursive digital filters with infinite impulse response (IIR) as well as numerical coefficients [14; and others]. Similar procedures have been offered within genetic algorithms (GAs) (see [15] for background) using numerical filter coefficients and time delays for analogy IIR filters [16], or powers-of-two coefficients for finite impulse response (FIR) discrete-time digital filters [17]. Modified GAs have also been used to design CMOS operational amplifiers (op amps) [18, 19]; however, the topology of each op amp was constrained to be 1 of 24 preselected topologies based on the conventional human-designed stages of an op amp. A GA was also used to evolve the topology of passive linear circuits composed of two-leaded components such as capacitors, resistors, and inductors [20]; however, component values were determined by subsequent numerical optimization.

Evolvable digital hardware [21-23] offers a potential approach to automating the synthesis of circuits. A GA has been used to evolve a frequency discriminator on a Xilinx XC6216 rapidly reconfigurable field programmable gate array operating in analog mode [24]. Also, Gruau [25] used cellular encoding and genetic programming to evolve the architecture, weights, thresholds, and biases of neurons in a neural network.

3. EIGHT PROBLEMS OF ANALOG CIRCUIT SYNTHESIS

The current effort concerns a suite of eight specific problems of analog circuit synthesis. The desired circuits comprise a variety of components, including transistors, diodes, resistors, inductors, and capacitors. The circuits have zero, one, or two inputs, and have one or two outputs. The eight problems involve designing:

1) A lowpass filter having a one-input, one-output circuit composed of capacitors and inductors that passes all frequencies below 1 kHz and suppresses all frequencies above 2 kHz. Filters are ubiquitous in analog electrical circuitry.

2) A crossover (woofer and tweeter) filter having a one-input, two-output circuit composed of similar components that passes all frequencies below 2.512 kHz to its first output port (the woofer), passes all higher frequencies to its second output port (the tweeter), and simultaneously suppresses the higher frequencies at the woofer and the lower frequencies at the tweeter.

3) A four-way source identification circuit having one input and one output and composed of resistors, capacitors, and inductors that produces an output of 1/3, 2/3, or 1 V for incoming signals whose frequencies are within 10% of 256 Hz, 750 Hz, or 2.56 kHz, respectively, but produces an output of 0 V otherwise.

4) A computational circuit having one input and one output and composed of transistors, diodes, resistors, and capacitors that produces an output voltage equal to the cube root of its input. The design of such circuits often relies on clever exploitation of the underlying device physics of the electrical components that is unique to the particular desired mathematical function [26-28]. Computational circuits are of special practical importance when the small number of required mathematical calculations does not warrant converting an analog signal into a digital signal, performing the mathematical function in the digital domain, and then converting the result back to the analog domain.

5) A time-optimal controller circuit having two inputs and one output composed of similar components to those used above that is to control the flight path of an aircraft with nonzero turning radius such that the aircraft flies (at constant speed and altitude) to an arbitrary destination in minimal time. The inputs are two voltages representing the two-dimensional location of the target point in the aircraft's frame of reference. The output is the aircraft's turn angle.

6) An amplifier composed of similar components and capable of delivering amplification of about 100 dB (i.e., 100,000 to 1) with low distortion, and little if any bias. Amplifiers are used for many purposes in analog electrical circuitry.

7) A temperature-sensing circuit that is composed of similar components and produces an output voltage proportional to the circuit's ambient temperature (between freezing and boiling). There is no explicit input, and one output. The circuit is to infer its ambient temperature from the fact that its own electrical components operate differently at different temperatures.

8) A voltage reference source circuit that is composed of similar components and delivers a constant 2 V in spite of variation in its incoming 5 V power supply voltage of ±1 V, and in spite of variation in the circuit's ambient temperature between freezing and boiling. The voltage supplied by all practical sources of electrical power are subject to variation, yet electrical circuits operate properly only when they are supplied with reference voltages that closely adhere to prespecified values. In addition, the behavior of electric circuits are subject to variation caused by the circuit's ambient temperature. Thus there is a need for a circuit that consistently supplies a prespecified constant voltage (within a small prespecified tolerance) in spite of variations in the incoming power supply and variations in the circuit's ambient temperature. The output is a

function of two variables (incoming voltage and temperature); however, the incoming power supply voltage is the only explicit input.

4. USING GENETIC PROGRAMMING TO EVOLVE CIRCUITS

The circuits are developed using genetic programming (GP) [29, 30], an extension of the GA in which the population consists of computer programs. Multipart programs consisting of a main program and one or more reusable, parametrized, hierarchically-called subprograms can be evolved using automatically defined functions (ADFs) [31, 32]. Architecture-altering operations [33] automatically determine the number of such subprograms, the number of arguments that each possesses, and the nature of the hierarchical references, if any, among such ADFs. See [34-37] for current research in GP.

The procedure breeds a population of rooted, point-labeled trees (acyclic graphs) with ordered branches, whereas electrical circuits consist of line-labeled cyclic graphs. GP can be applied to circuits if a mapping is established between the program trees used in GP and the cyclic graphs germane to electrical circuits.

The principles of developmental biology suggest a way to map program trees into circuits via a growth process that begins with a simple "embryonic circuit." This embryo may include the inputs and outputs of the particular circuit being designed and may also include certain fixed components (such as source and load resistors). The embryo also contains certain wires that are capable of subsequent modification. Until these are modified, however, the circuit does not produce any interesting output. An electrical circuit is progressively developed by applying various functions in a circuit-constructing program tree to the modifiable wires of the embryonic circuit (and as the circuit develops, also to the components and the modifiable wires of the successor circuits).

The functions in the circuit-constructing program trees are divided into four categories: 1) connection-modifying functions (CMFs) that alter the circuit topology, 2) component-creating functions (CCFs) that insert components into the circuit, 3) arithmetic-performing functions that appear in subtrees as argument(s) to the CCFs and specify the numerical value of the component, and 4) ADFs that appear in the function-defining branches and potentially enable certain substructures of the circuit to be reused (with parametrization).

Each branch of the program tree is created in accordance with a constrained syntactic structure. Branches are composed of construction-continuing subtrees that continue the developmental process and arithmetic-performing subtrees (APSs) that determine the numerical value of components. CMFs have one or more construction-continuing subtrees, but no APSs. CCFs have one or more construction-continuing subtrees and typically have one APS. This constrained syntactic structure is preserved using structure-preserving crossover with point typing (see [31]).

4.1 The Embryonic Circuit

In the automated process for circuit synthesis described here, an electrical circuit is created by executing a circuit-constructing program tree. The tree contains various component-creating and topology-modifying functions. Each tree in the population creates one circuit. The developmental process uses the program tree to convert an embryonic circuit into a fully developed circuit, and the specific embryo used depends on the number of inputs and outputs.

Figure 1 shows a one-input, one-output embryonic circuit in which VSOURCE is the input signal and VOUT is the output signal (the probe point). The circuit is driven by an incoming alternating circuit source VSOURCE. There is a fixed load resistor RLOAD and a fixed source

resistor RSOURCE in the embryo. In addition to the fixed components, there is a modifiable wire Z0 between nodes 2 and 3. All development originates from this modifiable wire.

[INSERT FIGURE 1]

4.2 Component-Creating Functions

Each program tree in the population contains CCFs and CMFs. The CCFs insert a component into the developing circuit and assign component value(s) to the component.

Each CCF has a writing head that points to an associated highlighted component in the developing circuit and modifies that component in a specified manner. The construction-continuing subtree of each CCF points to a successor function or terminal in the circuit-constructing program tree.

The APS of a CCF consists of a composition of arithmetic functions (addition and subtraction) and random constants (in the range -1.000 to +1.000). The APS specifies the numerical value of a component by returning a floating-point value that is interpreted on a logarithmic scale as the value for the component in a range of 10 orders of magnitude (using a unit of measure that is appropriate for the particular type of component [38,39]).

The two-argument resistor-creating R function causes the highlighted component to be changed into a resistor. The value of the resistor in kilo Ohms (k Ω) is specified by its APS (one of two arguments of the R function). The second argument is a construction-continuing subtree.

Fig. 2a shows a modifiable wire Z0 connecting nodes 1 and 2 of a partial circuit containing four capacitors. The circle indicates that Z0 has a writing head (i.e., is the highlighted component) so that Z0 is subject to subsequent modification.

Fig. 2b shows the result of applying the R function to the modifiable wire Z0 of Fig. 2a. The circle indicates that the newly created R1 has a writing head so that R1 remains subject to subsequent modification.

[INSERT FIGURES 2a and 2b]

Similarly, the two-argument capacitor-creating C function causes the highlighted component to be changed into a capacitor whose value in micro Farads (μF) is specified by its APS.

The one-argument Q_D_PNP diode-creating function causes a diode to be inserted in lieu of the highlighted component. This function has only one argument because there is no numerical value associated with a diode and thus no APS. In practice, the diode is implemented here using a pnp transistor whose collector and base are connected to each other. The Q_D_NPN function inserts a diode using an npn transistor in a similar manner.

There are also six one-argument transistor-creating functions (Q_POS_COLL_NPN, Q_GND_EMIT_NPN, Q_NEG_EMIT_NPN, Q_GND_EMIT_PNP, Q_POS_EMIT_PNP, Q_NEG_COLL_PNP) that insert a bipolar junction transistor in lieu of the highlighted component and that directly connect the collector or emitter of the newly created transistor to a fixed point of the circuit (the positive power supply, ground, or the negative power supply). These functions have only one argument because there is no numerical value associated with a transistor and thus no APS. There are only six such functions in this family because the other possibilities are not electronically reasonable.

Figure 3 shows the result of applying Q_POS_COLL_NPN that inserts a bipolar junction transistor whose collector is connected to the positive power supply. The function operates on the modifiable wire Z0 of Fig. 2a, thereby creating a transistor Q6 connecting nodes 1 and 2 in

Figure 3. The base of the transistor is connected to the node at the positive end of Z0 (node 2 of Fig. 2a) and the emitter is connected to the node at the negative end of Z0 (node 1 of Fig. 2a). The collector is connected to a +5 V DC positive power supply POS.

[INSERT FIGURE 3]

Each of the functions in the family of six different three-argument transistor-creating Q_3_NPN functions causes an npn bipolar junction transistor to be inserted in place of the highlighted component and one of the nodes to which the highlighted component is connected. The Q_3_NPN function creates five new nodes and three modifiable wires. There is no writing head on the new transistor, but there is a writing head on each of the three new modifiable wires. There are 12 members (called Q_3_NPN0, ..., Q_3_NPN11) in this family of functions because there are two choices of nodes (1 and 2) in Fig. 2a to be bifurcated and then there are six ways of attaching the transistor's base, collector, and emitter after the bifurcation. Figure 4 shows the result of applying the Q_3_NPN0 function, thereby creating transistor Q6 in lieu of the modifiable wire Z0 of Fig. 2a.

[INSERT FIGURE 4]

Similarly the family of 12 Q_3_PNP functions causes a pnp bipolar junction transistor to be inserted. For additional details of these functions see [38-45].

4.3 Connection-Modifying Functions

Each CMF in a program tree points to an associated highlighted component and modifies the topology of the developing circuit.

The one-argument polarity-reversing FLIP function attaches the positive end of the highlighted component to the node to which its negative end is currently attached and vice versa. After execution of the FLIP function, there is one writing head pointing to the component.

The three-argument SERIES division function creates a series composition consisting of the highlighted component (with a writing head), a copy of it (with a writing head), one new modifiable wire (with a writing head), and two new nodes. Fig. 5 illustrates the result of applying the SERIES division function to resistor R1 from Fig. 2b.

[INSERT FIGURE 5]

The four-argument PSS and PSL parallel division functions create a parallel composition consisting of the original highlighted component (with a writing head), a copy of it (with a writing head), two new modifiable wires (each with a writing head), and two new nodes. Fig. 6 shows the result of applying PSS to the resistor R1 from Fig. 2b.

[INSERT FIGURE 6]

There are six three-argument functions (T_GND_0, T_GND_1, T_POS_0, T_POS_1, T_NEG_0, T_NEG_1) that insert two new nodes and two new modifiable wires, and then make a connection to ground, positive power supply, or negative power supply, respectively. Fig. 7 shows the T_GND_0 function connecting resistor R1 of Fig. 2b to ground. There are two members in each of these three families of functions (T_GND_0 and T_GND_1 for the first family) because resistor R1 can become connected to node 1 or node 2.

[INSERT FIGURE 7]

There are two three-argument functions (PAIR_CONNECT_0 and PAIR_CONNECT_1) that enable distant parts of a circuit to be connected together. The first PAIR_CONNECT to occur in the development of a circuit creates two new wires, two new nodes, and one temporary port. The next PAIR_CONNECT creates two new wires and one new node, connects the temporary port to the end of one of these new wires, and then removes the temporary port.

The one-argument NOOP function has no effect on the highlighted component; however, it delays activity on the developmental path on which it appears in relation to other developmental paths in the overall program tree.

The zero-argument END function causes the highlighted component to lose its writing head, thereby ending that particular developmental path.

The zero-argument SAFE_CUT function causes the highlighted component to be removed from the circuit provided that the degree of the nodes at both ends of the highlighted component is three (i.e., no dangling components or wires are created).

5. PREPARATORY STEPS

Before applying GP to circuit synthesis, seven major preparatory steps are required: 1) identify the suitable embryonic circuit, 2) determine the architecture of the overall circuit-constructing program trees, 3) identify the terminals of the programs to be evolved, 4) identify the primitive functions contained in these programs, 5) create the fitness measure, 6) choose control parameters (e.g., population size, the maximum number of generations), and 7) determine the termination criterion and method of result designation. Each of these is discussed below.

5.1 Embryonic Circuit

The embryonic circuit used on a particular problem depends on the circuit's number of inputs and outputs. Fig. 1 showed an embryo suitable for a one-input, one output circuit. But consider a crossover (woofer and tweeter) filter having a single input and two outputs. This circuit has two probe points VOUT1 and VOUT2, not just one. Moreover, each probe point needs its own separate load resistor. Finally, there must be a way to establish connections between the input and the two outputs. Thus, the embryo of Fig. 1 is not suitable for the crossover filter.

Fig. 8 shows an embryonic circuit that meets the requirements of the problem of designing a crossover filter. The three modifiable wires Z0, Z1, and Z2 provide full connectivity between the single input at node 2 and the two outputs at nodes 3 and 6. This embryonic circuit has two probe points (VOUT1 and VOUT2) and two separate load resistors (RLOAD1 and RLOAD2, respectively).

[INSERT FIGURE 8]

All development originates from the modifiable wires of the embryonic circuit. For each such wire there is an associated result-producing branch in the circuit-constructing program tree. The top portion of Fig. 8 contains a highly abbreviated circuit-constructing program tree showing only the uppermost functions (L, C, and C, respectively) of the three result-producing branches and omitting the APSs associated with the L, C, and C functions. In the figure, the inductor-creating function L in the first result-producing branch points to the modifiable wire Z0. There is a writing head associated with each function in the program tree. When the L function is executed, it converts Z0 into an inductor; the newly created inductor acquires the writing head.

Similarly, the two capacitor-creating C functions convert modifiable wires Z1 and Z2 into capacitors (each with a writing head).

Note that a valid one-input, two output circuit is always produced when any sequence of CCFs and CMFs is applied to the embryo. For example, after the three wires are modified by the three CCFs in Fig. 8, connectivity is maintained between the original input at node 2 and the two original outputs at nodes 3 and 6. Similarly, connectivity will also be maintained between the original input and outputs when the topology of the developing circuit is later modified by CMFs. It should also be noted that the number of items connected to each node of the embryonic circuit is either two or three. Moreover, after the execution of any sequence of CCFs and CMFs, the number of items connected to each node of every fully developed circuit will also be either two or three. That is, the CCFs and CMFs preserve both circuit validity and the degree of connectivity (two or three) of each node.

The foregoing two embryonic circuits (and all of the subsequent embryos herein) can each be viewed as having two distinct parts. First, the inputs, outputs, and their associated source and load resistors can together be viewed as a fixed "harness" that does not change during the developmental process. Second, the modifiable wires provide an initial means of connecting the inputs to the outputs and provide a site for the development of the full circuit.

The problem of designing a time-optimal controller differs from the above two problems in that the desired controller must have two inputs and one output. Thus, its embryonic circuit (Fig. 9) has two inputs VSOURCE1 and VSOURCE2, and each input has a separate load resistor (RSOURCE1 and RSOURCE2, respectively). This embryo has three modifiable wires Z0, Z1, and Z2 that provide full connectivity between the two inputs at nodes 2 and 6 and the single output at node 3. All development originates from the three modifiable wires Z1, Z2, and Z3.

[INSERT FIGURE 9]

The desired temperature-sensing circuit has no explicit input and one output. Fig. 10 shows a suitable embryo. It has a loop consisting of two modifiable wires Z1 and Z2.

[INSERT FIGURE 10]

In some problems, such as designing an amplifier, the embryo contains additional fixed components because additional problem-specific functionality is required. Fig. 11 shows a one-input, one-output feedback embryo containing a 100 mega Ohm feedback resistor RFEEDBACK and a 100 Ω source resistor RSOURCE. The possible amplification of the circuit that will be evolved in lieu of the three modifiable wires Z0, Z1, and Z2 will be limited to 120 dB (a 1,000,000-to-1 gain) by the ceiling established by the ratio of the feedback resistor RFEEDBACK to the source resistor RSOURCE. This embryo also contains a 100 mega Ohm balance feedback resistor RBALANCE_FEEDBACK and a 100 Ω balance source resistor RBALANCE_SOURCE.

[INSERT FIGURE 11]

There is often considerable flexibility in choosing the embryonic circuit. For example, Fig. 1 showed the one-input, one-output embryo with one modifiable wire that was used for the source identification circuit, voltage reference circuit, and the computational circuit for the cube root;

however, an embryo with two modifiable wires (Z0 and Z1) (Fig. 12) would also have been suitable for these circuits and was, in fact, used for the lowpass filter.

[INSERT FIGURE 12]

5.2 Program Architecture

Since there is one result-producing branch in the program tree for each modifiable wire in the embryo, the architecture of each circuit-constructing program tree depends on the embryonic circuit. One result-producing branch was used for problems 3, 4, and 8, two were used for problems 1 and 7, and three were used for problems 2, 5, and 6 (see above). The architecture of each circuit-constructing program tree also depends on the use, if any, of ADFs. ADFs and architecture-altering operations were used in problems 3, 5, and 6. For these problems, each program in the initial population of programs had a uniform architecture with no ADFs. In later generations, the number of ADFs, if any, emerged as a consequence of the architecture-altering operations.

5.3 Function and Terminal Sets

The function set for each design problem depended on the type of electrical components that were used to construct the circuit. Capacitors, diodes, and transistors were used for five of the eight circuits (problems 4-8). Resistors were used for problem 3. When transistors were used, functions to provide connectivity to the positive and negative power supplies were also included, except that these were not included for problem 8 because the power supply was, by the nature of the problem, an independent input to the circuit.

For problems 4, 5, 6, and 7, the function set, $\mathcal{F}_{\text{ccs-initial}}$, for each construction-continuing subtree was

$\mathcal{F}_{\text{ccs-initial}}$ = {R, C, SERIES, PSS, PSL, FLIP, NOOP, T_GND_0, T_GND_1, T_POS_0, T_POS_1, T_NEG_0, T_NEG_1, PAIR_CONNECT_0, PAIR_CONNECT_1, Q_D_NPN, Q_D_PNP, Q_3_NPN0, ..., Q_3_NPN11, Q_3_PNP0, ..., Q_3_PNP11, Q_POS_COLL_NPN, Q_GND_EMIT_NPN, Q_NEG_EMIT_NPN, Q_GND_EMIT_PNP, Q_POS_EMIT_PNP, Q_NEG_COLL_PNP}.

For the npn transistors, the Q2N3904 model was used. For pnp transistors, the Q2N3906 model was used.

The initial terminal set, $\mathcal{T}_{\text{ccs-initial}}$, for each construction-continuing subtree was

$\mathcal{T}_{\text{ccs-initial}}$ = {END, SAFE_CUT}.

The initial terminal set, $\mathcal{T}_{\text{aps-initial}}$, for each arithmetic-performing subtree consisted of

$\mathcal{T}_{\text{aps-initial}}$ = {←},

where ← represents floating-point random constants from –1.0 to +1.0.

The function set, $\mathcal{F}_{\text{aps}}$, for each APS was,

$\mathcal{F}_{\text{aps}}$ = {+, -}.

The terminal and function sets were identical for all result-producing branches for a particular problem.

For problems 1-3 there was no need for functions to provide connectivity to the positive and negative power supplies.

For problems 3, 5, and 6 the architecture-altering operations were used and the set of potential new functions, $\mathcal{F}_{\text{potential}}$, was

$\mathcal{F}_{\text{potential}}$ = {ADF0, ADF1, ...}.

The set of potential new terminals, $\mathcal{T}_{potential}$, for the ADFs was
$\mathcal{T}_{potential}$ = {ARG0}.

The architecture-altering operations changed the function set, $\mathcal{F}_{ccs}$ for each construction-continuing subtree of all three result-producing branches and the function-defining branches, so that
$\mathcal{F}_{ccs} = \mathcal{F}_{ccs\text{-}initial} \approx \mathcal{F}_{potential}$.

The architecture-altering operations generally changed the terminal set for ADFs, $\mathcal{T}_{aps\text{-}adf}$, for each APS, so that
$\mathcal{T}_{aps\text{-}adf} = \mathcal{T}_{aps\text{-}initial} \approx \mathcal{T}_{potential}$.

5.4 Fitness Measure

The fitness measure guides the evolutionary process. The evaluation of each individual circuit-constructing program tree in the population begins with its execution. This execution applies the functions in each program tree to an embryonic circuit, thereby creating a fully developed circuit. A netlist is created that identifies each component of the developed circuit, the nodes to which each component is connected, and the value of each component. The netlist becomes the input to the 217,000-line SPICE (Simulation Program with Integrated Circuit Emphasis) simulation program [45]. SPICE then determines the behavior of the circuit. It was necessary to make considerable modifications in SPICE so that it could run as a submodule within the GP system.

Fig. 13 provides additional detail on the calculation of fitness. First, the current circuit is set to the embryonic circuit. The individual circuit-constructing program is then executed causing the CCFs and CMFs in the program tree to be applied. When completed, the current circuit is translated into a netlist.

[INSERT FIGURE 13]

The netlist is then simplified. All wires are removed and the nodes to which they are connected are merged. Dangling components and any isolated substructures are removed. A very large resistance (e.g., a 1 giga-Ohm resistor) is inserted between ground and any node for which there is no DC path to ground (e.g., a node to which only capacitors are connected). This has no significant electrical effect; however, it enables SPICE to numerically simulate the circuit. In addition, the time required for a SPICE simulation generally increases nonlinearly as a function of the number of nodes in the netlist (in an approximately subquadratic to quartic way). To shorten the netlist, all series and parallel compositions of like passive components are replaced, for purposes of the simulation only, by a single component of appropriate value.

The fitness measure varies for each problem. The high-level statement of desired circuit behavior is translated into a well-defined measurable quantity that can be used by GP to guide the evolutionary process. The specific fitness measures were as follows:

1) Lowpass filter: The desired lowpass LC filter should have a passband below 1 kHz and a stopband above 2 kHz. The circuit will be driven by an incoming AC voltage source with a 2 V amplitude. If the source (internal) resistance RSOURCE and the load resistance RLOAD in the embryonic circuit are each 1 k Ω, the incoming 2 V signal is divided in half.

The *attenuation* of the filter is defined in terms of the maximum signal in its stopband relative to the reference voltage (half of 2 V here). A *decibel* is a unitless measure of relative voltage that is defined as 20 times the common (base 10) logarithm of the ratio between the voltage at a particular probe point and a reference voltage. Since the maximum acceptable signal

in the stopband relative to the 1 V reference voltage is 1,000-to-1, the design goal calls for an attenuation of at least 60 dB.

In this problem, a voltage in the passband of exactly 1 V and a voltage in the stopband of exactly 0 V is regarded as ideal. The (preferably small) variation within the passband is called the *passband ripple*. Similarly, the incoming signal is never fully reduced to zero in the stopband of an actual filer. The (preferably small) variation within the stopband is called the *stopband ripple*. A voltage in the passband of between 970 mV and 1 V (i.e., a passband ripple of 30 mV or less) and a voltage in the stopband of between 0 V and 1 mV (i.e., a stopband ripple of 1 mV or less) is regarded as acceptable. Any voltage lower than 970 mV in the passband and any voltage above 1 mV in the stopband is regarded as unacceptable.

A practicing electrical engineer would recognize that the above design goals can be satisfied by a fifth-order *elliptic filter* (*Cauer filter*) with a modular angle Θ of 30 degrees (i.e., the arcsin of the ratio of the boundaries of the passband and stopband) and a reflection coefficient ρ of 20% [46].

Since the high-level statement of behavior for the desired circuit is expressed in terms of frequencies, the voltage VOUT is measured in the frequency domain. SPICE performs an AC small signal analysis and report the circuit's behavior for frequencies chosen over five decades (between 1 Hz and 100 kHz). Each decade is divided into 20 parts (using a logarithmic scale) so there are 101 fitness cases for this problem.

Fitness is measured in terms of the sum over these cases of the absolute weighted deviation between the actual value of the voltage that is produced by the circuit at the probe point VOUT and the target value for voltage. The smaller the value of fitness, the better. A fitness of zero represents an (unattainable) ideal filter.

Specifically, the standardized fitness is

$$F(t) = \sum_{i=0}^{100} \left[W(d(f_i), f_i) d(f_i) \right]$$

where f_i is the frequency of fitness case i; $d(x)$ is the absolute value of the difference between the target and observed values at frequency x; and $W(y,x)$ is the weighting for difference y at frequency x.

The fitness measure is designed to not penalize ideal values, to slightly penalize every acceptable deviation, and to heavily penalize every unacceptable deviation. Specifically, the procedure for each of the 61 points in the 3-decade interval between 1 Hz and 1 kHz is as follows: If the voltage equals the ideal value of 1.0 V in this interval, the deviation is 0.0. If the voltage is between 970 mV and 1 V, the absolute value of the deviation from 1 V is weighted by a factor of 1.0. If the voltage is less than 970 mV, the absolute value of the deviation from 1 V is weighted by a factor of 10.0.

The procedure for each of the 35 points in the interval from 2 kHz to 100 kHz similarly weights the acceptable and unacceptable deviations.

For each of the five points in the interval from 1 kHz to 2 kHz (i.e., the five points in the "don't care" band), the deviation is deemed to be zero.

The number of “hits” for this problem (and all other problems herein) is defined as the number of fitness cases for which the voltage is acceptable or ideal or that lie in the "don't care" band.

Many of the random initial circuits and many that are created by the crossover and mutation operations in subsequent generations cannot be simulated by SPICE. These circuits receive a

high penalty value of fitness (10^8) and become the worst-of-generation programs for each generation. For additional details, see [38, 39].

2) Crossover Filter: The main differences between the problem of designing the lowpass filter and the problem of designing the crossover filter are that the crossover filter has two outputs and that the tweeter output calls for a highpass filter (also fifth-order elliptic). The fitness measure for this problem reflects these requirements. SPICE is called to perform an AC small signal analysis and to report the circuit's behavior at two probe points, VOUT1 and VOUT2, instead of just one. Specifically, the standardized fitness, $F(t)$, is

$$F(t) = \sum_{i=0}^{100} \left[W_1(d_1(f_i), f_i) d_1(f_i) + W_2(d_2(f_i), f_i) d_2(f_i) \right]$$

where f_i is the frequency of fitness case i; $d_1(x)$ is the difference between the target and observed values at frequency x for probe point VOUT1; $d_2(x)$ is the difference between the target and observed values at frequency x for probe point VOUT2; $W_1(y,x)$ is the weighting for difference y at frequency x for probe point VOUT1; and $W_2(y,x)$ is the weighting for difference y at frequency x for probe point VOUT2. For additional details, see [40].

3) Source Identification Circuit: As before, fitness is measured in terms of the sum, over 101 fitness cases, of the absolute weighted deviation between the actual value of the voltage that is produced by the circuit and the target value for voltage.

The three points that are closest to the band located within 10% of 256 Hz are 229.1 Hz, 251.2 Hz, and 275.4 Hz. The procedure for each of these three points is as follows: If the voltage equals the ideal value of 0.5 V in this interval, the deviation is 0.0. If the voltage is more than 240 mV from 0.5 V, the absolute value of the deviation from 0.5 V is weighted by a factor of 20. If the voltage is more than 240 mV of 0.5 V, the absolute value of the deviation from 0.5 V is weighted by a factor of 200. This arrangement reflects the fact that the ideal output voltage for this range of frequencies is 0.5 V, that a 240 mV discrepancy is acceptable, and that a larger discrepancy is not acceptable.

Similar weighting was used for the three points (2.291 kHz, 2.512 kHz, and 2.754 kHz) that are closest to the band located within 10% of 2.560 kHz and for the three points (2.291 kHz, 2.512 kHz, and 2.754 kHz) that are closest to the band located within 10% of 750 Hz.

The procedure for each of the remaining 92 points is as follows: If the voltage equals the ideal value of 0 V, the deviation is 0.0. If the voltage is within 240 mV of 0 V, the absolute value of the deviation from 0 V is weighted by a factor of 1.0. If the voltage is more than 240 mV from 0 volts, the absolute value of the deviation from 0 V is weighted by a factor of 10. For additional details, see [47, 48].

4) Computational Circuit: The target voltage is the cube root of the input voltage. SPICE is called to perform a DC sweep analysis at 21 equidistant voltages between –250 mV and +250 mV. Fitness is the sum, over these 21 fitness cases, of the absolute weighted deviation between the actual value of the voltage that is produced by the circuit and the target value for voltage. Also see [49].

5) Time-Optimal Controller Circuit: The fitness of a controller was evaluated using 72 randomly chosen fitness cases each representing a different target (fly-to) point. Fitness is the sum, over the 72 fitness cases, of the fly-to times. If the aircraft came within a capture radius of 0.28 nautical miles (NM) of its target point before the end of the 80 time steps allowed for a particular fitness case, the contribution to fitness for that fitness case was the actual fly-to time. However, if the aircraft failed to come within the capture radius during the 80 time steps, the contribution to fitness was 0.160 hours (i.e., double the worst possible time).

SPICE performs a nested DC sweep, which provides a way to simulate the DC behavior of a circuit with two inputs. It resembles a nested pair of FOR loops in a computer program in that both of the loops have a starting value for the voltage, an increment, and an ending value for the voltage. For each voltage value in the outer loop, the inner loop simulates the behavior of the circuit by stepping through its range of voltages. Specifically, the starting value for voltage is –4 V, the step size is 0.2 V, and the ending value is +4 V. These values correspond to the dimensions of the aircraft's world of 64 square nautical miles (NM) extending 4 NM in each of the four directions from the origin of a coordinate system (i.e., 1 V equals 1 NM).

When an individual program is executed, it produces a numeric value which the wrapper (output interface) transforms into the turn angle Θ for the aircraft. Also see [50].

6) Amplifier: SPICE was requested to perform a DC sweep analysis to determine the circuit's response for several different DC input voltages. An ideal inverting amplifier circuit would receive the DC input, invert it, and multiply it by the amplification factor. A circuit is flawed to the extent that it does not achieve the desired amplification, the output signal is not perfectly centered on 0 V (i.e., it has a bias), or the DC response is not linear. The circuits are analyzed with a 5 point DC sweep ranging from –10 mV to +10 mV, with input points at –10 mV, –5 mV, 0 mV, +5 mV, and +10 mV.

Fitness is then calculated by summing an amplification penalty, a bias penalty, and two non-linearity penalties – each derived from these five DC outputs.

The amplification factor of the circuit is measured by the slope of the straight line between the output for –10 mV and the output for +10 mV (i.e., between the outputs for the endpoints of the DC sweep). If the amplification factor is less than the maximum allowed by the feedback resistor (120 dB for this problem), there is a penalty equal to the shortfall in amplification.

The bias is computed using the DC output associated with a DC input of 0 V. The penalty is equal to the bias times a weight of 0.1.

Linearity is measured by the deviation between the slope of each of two line segments and the overall amplification factor of the circuit. The first line segment spans the output values associated with inputs of –10 mV through –5 mV. The second line segment spans the output values associated with inputs of +5 mV through +10 mV. The penalty for each of these line segments is equal to the absolute value of the difference in slope between the respective line segment and the overall amplification factor of the circuit. Also see [51].

7) Temperature-Sensing Circuit: The circuit will operate in an environment whose temperature ranges between 0˚ C (freezing) and 100˚ C (boiling) and will be expected to report the ambient temperature in voltages using a linear scale in which 0 V equals 0˚ C and 10 V equals 100˚ C. SPICE performs 21 consecutive operating point analyses for temperatures between 0˚ and 100˚ C in increments of 5˚ C.

Fitness is the sum over the 21 fitness cases of the absolute value of the weighted difference between the output voltage and the target voltage.

If the output voltage is within 0.1 V of the target, the absolute value of the deviation is weighted by a factor of 1.0. If the deviation is greater, the absolute value of the deviation is weighted by a factor of 10.0.

8) Voltage Reference Circuit: The goal is to design a voltage reference source that delivers a prespecified constant 2 V (with a small prespecified tolerance) in spite of variation in the voltage of the incoming 5 V power supply of ± 20% (i.e., from 4 to 6 V) and variation in the circuit's ambient temperature between 0˚ and 100˚ C. This one-output circuit has only one input (the incoming power supply voltage).

SPICE performs five consecutive DC sweeps. Each is performed for a particular temperature: 0˚, 25˚, 50˚, 75˚, and 100˚ C. Each sweep runs from 4 to 6 V in 21 increments of 0.1 V.

Fitness is the sum, over the 105 fitness cases, of the absolute value of the weighted difference between the output voltage and the target voltage.

If the output voltage is within the acceptable range of 0.1 volts of the target, the absolute value of the deviation is weighted by a factor of 1.0. If the deviation is greater, the absolute value of the deviation is weighted by a factor of 10.0.

5.5 Control Parameters

The population size, *M*, was 640,000 for all problems.

For problems 1, 2, 4, 7, and 8 the percentages of the genetic operations [29] on each generation were the same as those used over a period of years on numerous other problems that the authors have solved using genetic programming. Specifically, the percentages were 89% one-offspring crossovers, 10% reproductions, and 1% mutations. No effort has been made to determine the sensitivity of these parameters.

For the problems on which ADFs and architecture-altering operations were used (i.e., problems 3, 5, and 6), the percentages of operations on each generation were again based on the author's consistent prior practice using ADFs and architecture-altering operations. Specifically, the percentages after generation 5 were 86.5% one-offspring crossovers, 10% reproductions, 1% mutations, 1% branch duplications, 0.5% branch deletions, and 1% branch creations. Since we did not want to waste large amounts of computer time in early generations where only a few programs have any ADFs at all, the percentage of operations on each generation before generation 6 was 78.0% one-offspring crossovers, 10% reproductions, 1% mutations, 5.0% branch duplications, 1% branch deletions, and 5.0% branch creations.

The other parameters were substantially the same for each of the eight problems (as presented in the individual references cited above for each problem).

Each problem was run on a medium-grained parallel Parsytec computer system [52] consisting of 64 80-MHz PowerPC 601 processors arranged in an 8 by 8 toroidal mesh with a host PC Pentium type computer. The distributed GA was used with a population size of Q = 10,000 at each of the D = 64 demes (reproductive populations). On each generation, four boatloads of emigrants, each consisting of B = 2% (the migration rate) of the node's subpopulation (selected on the basis of fitness) were dispatched to each of the four toroidally adjacent processing nodes.

5.6 Termination Criterion and Results Designation

Each run was monitored and terminated when it was in substantial compliance with all the design requirements of the problem. The best-so-far individual was designated as the result of each run.

6. RESULTS

In all eight problems, fitness was observed to improve over successive generations. Satisfactory results were generated in every case on the first or second trial. (In most cases on the first trial. Moreover, when two runs were required, the first produced a nearly satisfactory result.) This rate of success suggests that the capabilities of the approach and current computing system have not been fully exploited. In addition, although a large majority of the initial circuits were not able to be simulated by SPICE, most were simulatable after only a few generations. This suggests that the genetic operators have the property that offspring from simulatable circuits are also able to be simulated. Specific results for each problem are listed below.

6.1 Lowpass Filter

Many of the runs produced lowpass filters having a topology similar to that employed by human engineers. For example, in generation 32 of one run, a circuit (Fig. 14a) was evolved with a near-zero fitness of 0.00781. The circuit was 100% compliant with the design requirements in that it scored 101 hits (out of 101). After the evolutionary run, this circuit (and all evolved circuits herein) were simulated anew using the commercially available Microsim circuit simulator to verify performance. This circuit had the recognizable ladder topology [46] of a Butterworth or Chebychev filter (i.e., a composition of series inductors horizontally with capacitors as vertical shunts).

[INSERT FIGURES 14a and 14b]

Fig. 14b shows the behavior in the frequency domain of this genetically evolved lowpass filter. As can be seen, the genetically evolved circuit delivers around a full volt for all frequencies up to 1 kHz and essentially 0 V for all frequencies above 2 kHz.

In another run, a recognizable "bridged T" arrangement (Fig. 15a) was evolved with a near-zero fitness. The "T" consists of capacitors C3 and C15 and inductor L11 and the "bridge" consists of inductor L14. The frequency domain behavior of this 100%-compliant circuit is indistinguishable from that of Fig. 14b.

In another run using ADFs [43], a 100% compliant circuit with the recognizable elliptic (Cauer) topology was evolved (Fig. 15b). This genetically evolved circuit has the equivalent of six inductors horizontally across the top of the circuit and five vertical shunts. Each shunt consists of an inductor and a capacitor (e.g., L34 and C18 appear in the first shunt). This is the topology of the elliptic invented and patented by Cauer. The Cauer filter was a significant advance (both theoretically and commercially) over the Butterworth and Chebychev filters. For example, for one illustrative set of commercial specifications, a 5th-order elliptic filter could match the performance of 17th-order Butterworth filter or an 8th-order Chebychev filter. The 8th-order Chebychev filter has one more component than the 5th-order elliptic filter.

[INSERT FIGURES 15a AND 15b]

Thus, genetic programming rediscovered the ladder topology used in Butterworth and Chebychev filters, the "bridged T" topology, and the elliptic (Cauer) topology.

6.2 Crossover Filter

The evolved crossover (woofer and tweeter) filter consists of a recognizable combination of a standard lowpass ladder filter and an almost standard highpass ladder filter. The lowpass part of this circuit from generation 137 (Fig. 16a) has the ladder topology. Except for additional capacitor C38, the highpass part also has the ladder topology. Note that the locations of the capacitors and inductors in the ladder topology are reversed between the lowpass and highpass filters.

[INSERT FIGURES 16a AND 16b]

Fig. 16b shows the behavior in the frequency domain of this genetically evolved crossover filter. This circuit has a fitness of 0.7807 and scores 192 hits (out of 202). The two outputs crossover in the desired area around 2.512 kHz. The performance of this circuit is slightly better than the combination of lowpass and highpass Butterworth filters of order 7.

6.3 Source Identification Circuit

The genetically evolved crossover four-way source identification circuit from generation 85 achieves a fitness of 404.3 and scores a total of 199 hits (out of 202). Fig. 17a shows this circuit after expansion of its ADFs.

[INSERT FIGURES 17a AND 17b]

Fig. 17b shows the behavior of this best-of-run circuit in the frequency domain. The three boxes indicate the range of allowable output voltages (along the linear vertical axis) for the specified range of input frequencies (along the logarithmic horizontal axis). As can be seen, the circuit produces the desired 1 V, 2/3 V, and 1/3 V (each within the 1/6 V tolerance) for the three specified narrow bands of frequencies and the desired near-zero signal for all other frequencies.

6.4 Computational circuit

The genetically evolved computational circuit for the cube root from generation 60 (Fig. 18a), achieves a fitness of 1.68, and has 36 transistors, two diodes, no capacitors, and 12 resistors (in addition to the source and load resistors in the embryo).

[INSERT FIGURES 18a AND 18b]

Fig. 18b shows the output voltage produced by this best-of-run circuit (along the linear vertical axis) overlain over the target values (i.e., the cube root of the input voltage). The two curves are almost indistinguishable.

6.5 Time-Optimal Controller Circuit

The best-of-run time-optimal controller circuit (Fig. 19) appeared in generation 31, scores 72 hits, and achieves a near-optimal fitness of 1.541 hours. In comparison, the optimal value of fitness for this problem is known to be 1.518 hours [53]. This best-of-run circuit has 10 transistors and 4 resistors. The program has one ADF that is called twice.

[INSERT FIGURE 19]

6.6 Amplifier

The best circuit from generation 86 (Fig. 20a) achieves a fitness of 938,427.3. Based on the DC sweep, the amplification is 96.2 dB (64,860-to-1) and the bias is 7.44 V. Based on a transient analysis at 1,000 Hz, the amplification is 94.1 dB; the bias is 7.46 V; and the distortion is 7.07%. The 3 dB bandwidth is 1,078.4 Hz. The program has two ADFs. `ADF0` is called once; however, `ADF1` is not called. The circuit has 25 transistors, no diodes, two capacitors, and two resistors (in addition to the five resistors of the feedback embryo). Thus, a high-gain amplifier with low distortion and acceptable bias has been evolved.

[INSERT FIGURES 20a AND 20b]

Fig. 20b shows `ADF0` (with 12 transistors, no diodes, one capacitor, and two resistors).

6.7 Temperature-Sensing Circuit

The best temperature-sensing circuit from generation 25 (Fig. 21a) achieves a fitness of 26.4, scores 16 hits (out of 21), and is composed of 42 transistors, six diodes, and six resistors (not counting the one fixed resistor inherited from the embryonic circuit).

[INSERT FIGURES 21a AND 21b]

Fig. 21b shows the output voltage (along the linear vertical axis) of the best evolved temperature-sensing circuit as a function of temperature. The circuit provides a good measure of temperature.

6.8 Voltage Reference Circuit

We illustrate the progressive nature of the evolutionary process by showing the best-of-generation circuits for several selected generations of the run.

The best circuit from generation 0 (Fig. 22a) achieves a fitness of 131.1, scores 10 hits, and is composed of three transistors, two diodes, and one resistor (not counting the two fixed resistors inherited from the embryonic circuit).

[INSERT FIGURES 22a AND 22b]

Fig. 22b shows the voltages produced by the best circuit of generation 0. The top graph indicates the voltages produced for 0˚ C when the incoming 5 V power supply actually delivers between 4 and 6 V. The four remaining graphs in the figure represent 25˚, 50˚, 75˚, and 100˚ C.

The four panels of Fig. 23 show the output voltage (along the linear vertical axis) produced by the best circuit of generations 0, 6, 49, and 80. As one proceeds over these four selected generations, fitness improves (decreases) from 131.1, 65.7, 35.2, and 35.2, respectively, as shown by the progressive coalescence of the five graphs around the desired output of 2 V and a progressive flattening of the five graphs. The best circuit from generation 80 delivers almost constant voltages.

[INSERT 4 PANELS OF FIGURE 23]

The best circuit (Fig. 24) from generation 6 achieves a fitness of 65.7, scores 16 hits (out of 105), and has six transistors, two diodes, and two resistors.

[INSERT FIGURE 24]

The best circuit (Fig. 25) from generation 49 achieves a fitness of 35.2, scores 49 hits, and has 11 transistors, four diodes, and four resistors.

[INSERT FIGURE 25]

The best circuit (Fig. 26) from generation 80 achieves a fitness of 6.6, scores 90 hits (out of 105), and has 40 transistors, 14 diodes, and 13 resistors.

[INSERT FIGURE 26]

6.9 Other Circuits

Numerous other analog electrical circuits have been evolved using the above approach, including other amplifiers [44], a different source identification circuit [47], a comb filter [43], and an asymmetric bandpass filter with stringent requirements [41].

7. CONCLUSION

We have demonstrated that genetic programming can automatically design the topology and sizing of eight different prototypical analog electrical circuits containing various components and having various numbers of inputs and outputs. There is no previously known general automated technique for synthesizing an analog electrical circuit from a high-level statement of the circuit's desired behavior. The genetically evolved circuits for the eight problems constitute an instance of an evolutionary computation technique producing results on a task that is usually thought of as requiring human intelligence. The problem-specific information required for each of the eight problems was minimal and consisted primarily of the number of inputs and outputs of the desired circuit, the types of available components, and a fitness measure that restates the high-level statement of the circuit's desired behavior as a measurable mathematical quantity. The fact that a uniform approach was taken to each of the design problems studied here, along with the satisfactory results that were obtained in at most two runs of genetic programming, suggests that genetic programming is generally applicable to the problem of synthesis of analog electrical circuits.

The question arises as to what would be required to extend the foregoing examples of the automated synthesis of analog electrical circuits to the design of commercial-quality circuits in the next few years. For specificity, this question can be posed in terms of designing an operational amplifier with characteristics equivalent to a commercial amplifier such as the μ741 op amp (whose design took many years). Addressing this question involves considering 1) the amount of computer time used to evolve the eight circuits presented herein, 2) the ever-increasing availability of computer time over the next few years, 3) optimizations for accelerating the process, and 4) the additional computing effort required to satisfy the stringent electrical requirements of a commercial-quality circuit such as the μ741 op amp.

The best-of-run circuits for the eight problems presented herein emerged between generations 31 and 137 (the average being 67). With a population size of 640,000 (with 10% Darwinian reproduction on each generation), the average result thus required 38,592,000 fitness evaluations. Each fitness evaluation required a separate SPICE simulation consisting of an average of 2.3×10^7 computer operations and consuming about 0.25 second of computer time on an 80-MHz processor. The runs for the eight problems took about 2 days each on a parallel computing system with 64 80-MHz processors and thus entailed about 10^{15} operations each. Petaflop computers [54] capable of executing 10^{15} operations per second are expected to be available to high-end institutional users by 2010. Thus, it will be possible to run the foregoing examples of automated synthesis in a matter of seconds in the relatively near future.

Even today, a single workstation of current vintage with dual 240-MHz processors could perform the above-mentioned 10^{15} operations in 21 days. If available computer capacity continues to double approximately every 18 months in accordance with Moore's law, this 21-day computation would, in six years, require only about 1.3 days.

None of the above computer times reflect known optimizations that could be implemented currently. For example, SPICE can be accelerated by a factor of three to ten by using improved versions of SPICE that are currently commercially available (as opposed to the public domain

SPICE simulator [45] that we modified and used). In addition, SPICE simulations can be accelerated by several fold [55] by using known techniques to tune its various control parameters to the specific type of circuit being designed. If the combined effect of both of these optimizations were an acceleration of 10-to-1, a single workstation of current vintage with dual 240-MHz processors could today execute the above-mentioned average run in about 2.1 days. Again, if available computer capacity continues to double approximately every 18 months, this 2.1-day computation would, in six years, require only about three hours.

The fitness measures for each of the eight problems incorporated the most important desired characteristics of the circuits and establish the principle that non-trivial analog circuits can be designed with genetic programming. However, the fitness measures for some of the problems did not incorporate certain additional secondary characteristics that would be required of commercial-quality circuits. For example, the fitness measure for the amplifier (problem 6) explicitly incorporated three important characteristics: gain, bias, and distortion. As it happened, the genetically evolved circuit generalized beyond these characteristics in the sense that it also possessed several unrequested desirable characteristics (e.g., reasonable bandwidth and parts count). However, the data sheet for a commercial-quality op amp includes several additional requirements (e.g., power supply rejection ratio, power consumption, slew rate) and various different component models may be used in particular amplifiers. Our ongoing work has already demonstrated the principle that genetic programming can evolve amplifiers that partially satisfy certain additional requirements. It is not known how much additional computational effort might be required to evolve a circuit that simultaneously satisfies all the requirements of a commercial quality op amp. We view creation of such a design as an appropriate challenge problem for this field.

ACKNOWLEDGMENTS

David B. Fogel, the reviewers, and Simon Handley made helpful comments on drafts of this paper. Dr. Joseph Babanezhad provided helpful information to Frank Dunlap concerning the computational circuits described in this paper. The authors worked with Jason Lohn on the computational circuits and the source identification circuits [47 - 49].

REFERENCES

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development,* 3(3) 210–229, July 1959.

[2] R. A. Rutenbar, "Analog design automation: Where are we? Where are we going?" *Proceedings of the l5th IEEE CICC*. New York: IEEE, 13.1.1-13.1.8, 1993.

[3] O. Aaserud and I. R. Nielsen, "Trends in current analog design: A panel debate." *Analog Integrated Circuits and Signal Processing*. 7(1) 5-9, 1995.

[4] G. J. Sussman, and R. M. Stallman, "Heuristic techniques in computer-aided circuit analysis." *IEEE Transactions on Circuits and Systems*, 22(11), November 1975.

[5] R. Harjani, R. A. Rutenbar, and L. R. Carley, "A prototype framework for knowledge-based analog circuit synthesis," *Proceedings of the 24th Design Automation Conference*. New York, NY: Association for Computing Machinery, 1987.

[6] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Transactions on Computer Aided Design*, 8:1247–1266, 1989.

[7] H. Y. Koh, C. H. Sequin, and P. R. Gray, "Automatic synthesis of operational amplifiers based on analytic circuit models," *Proceedings of IEEE International Conference on Computer-Aided Design*, November 1987.

[8] Koh, H. Y., Sequin, C. H. and Gray, P. R. 1990. OPASYN: A compiler for MOS operational amplifiers. *IEEE Transactions on Computer Aided Design*. 9:113–125.

[9] E. S. Ochotta, *Synthesis of High-Performance Analog Cells in ASTRX/OBLX*, Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1994.

[10] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(3) 273–294, 1996.

[11] W. E. Jones III, *Simultaneous Topology Selection and Sizing/Biasing for Analog Synthesis in ASTRX/OBLX*, Master's Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.

[12] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*, New York: John Wiley, 1966.

[13] I. Rechenberg, *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishments, Library Translation 1112, Farnborough, 1965.

[14] T. Gorne and M. Schneider, "Design of digital filters with evolutionary algorithms," In Albrecht, R. F., Reeves, C. R., and Steele, N. C. (editors), *Artificial Neural Nets and Genetic Algorithms*, Vienna: Springer-Verlag, pp. 368–374, 1993.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[16] A. Neubauer, "Genetic design of analog IIR filters with variable time delays for optically controlled microwave signal processors," *Proceedings of the First IEEE Conference on Evolutionary Computation,* IEEE Press, vol. I. pp. 437–442, 1994.

[17] P. Gentilli, F. Piazza, F., and A. Uncini, "Evolutionary design of FIR digital filers with power-of-two coefficients," *Proceedings of the First IEEE Conference on Evolutionary Computation,* IEEE Press, vol. I. pp. 110–114, 1994.

[18] M. W. Kruiskamp, *Analog Design Automation using Genetic Algorithms and Polytopes*, Eindhoven, The Netherlands: Data Library Technische Universiteit Eindhoven, 1996.

[19] M. W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS opamp synthesis by means of a genetic algorithm," *Proceedings of the 32nd Design Automation Conference*, New York, NY: Association for Computing Machinery, pp. 433–438, 1995.

[20] J. B. Grimbleby, "Automatic analogue network synthesis using genetic algorithms," *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, London: Institution of Electrical Engineers, pp. 53 – 58, 1995.

[21] Higuchi, Tetsuya, Niwa, Tatsuya, Tanaka, Toshio, Iba, Hitoshi, de Garis, Hugo, and Furuya, Tatsumi. 1993a. In Meyer, Jean-Arcady, Roitblat, Herbert L. and Wilson, Stewart W. (editors). *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: The MIT Press. 1993. pp. 417 – 424.

[22] Mizoguchi, Junichi, Hemmi, Hitoshi, and Shimohara, Katsunori. 1994. Production genetic algorithms for automated hardware design through an evolutionary process. *Proceedings of the First IEEE Conference on Evolutionary Computation.* IEEE Press. vol. I. pp. 661-664.

[23] E. Sanchez, and M. Tomassini, (editors), *Towards Evolvable Hardware*. Lecture Notes in Computer Science, vol. 1062, Berlin: Springer-Verlag, 1996.

[24] A. Thompson, "Silicon evolution," In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (editors), *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press, pp. 444–452, 1996.

[25] F. Gruau, "Artificial cellular development in optimization and compilation," In E. Sanchez, and M. Tomassini, (editors), *Towards Evolvable Hardware*. Lecture Notes in Computer Science, vol. 1062, Berlin: Springer-Verlag, pp. 48 – 75, 1996.

[26] B. Gilbert, "A precise four-quadrant multiplier with subnanosecond response," *IEEE Journal of Solid-State Circuits*, vol. SC-3, Number 4, pp. 365–373, December 1968.

[27] D. H. Sheingold, *Nonlinear Circuits Handbook*. Norwood, MA: Analog Devices Inc., 1976.

[28] J. N. Babanezhad, G. C. and Temes, "Analog MOS Computational Circuits," *Proceedings of the IEEE Circuits and System International Symposium*," pp. 1156–1160, 1986.

[29] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* Cambridge, MA: MIT Press, 1992.

[30] J. R. Koza, and J. P. Rice, *Genetic Programming: The Movie*, Cambridge, MA: MIT Press, 1992.

[31] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press, 1994.

[32] J. R. Koza, *Genetic Programming II Videotape: The Next Generation*. Cambridge, MA: MIT Press, 1994.

[33] J. R. Koza, "Evolving the architecture of a multi-part program in genetic programming using architecture-altering operations," In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel (editors), *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, Cambridge, MA: The MIT Press, pp. 695–717, 1995.

[34] K. E. Kinnear, Jr. (editor) *Advances in Genetic Programming*. Cambridge, MA: The MIT Press, 1994.

[35] P. J. Angeline, and K. E. Kinnear, Jr. (editors), *Advances in Genetic Programming 2*, Cambridge, MA: The MIT Press, 1996.

[36] J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (editors), *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press, 1996.

[37] J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo (editors), *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13–16, 1997, Stanford University*, San Francisco, CA: Morgan Kaufmann, 1997.

[38] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Toward evolution of electronic animals using genetic programming," *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA: The MIT Press, pp. 327-334, 1996.

[39] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming" In J. S. Gero and F. Sudweeks (editors), *Artificial Intelligence in Design '96*, Dordrecht: Kluwer, pp. 151-170, 1996.

[40] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Four problems for which a computer program evolved by genetic programming is competitive with human performance," *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 1–10, 1996.

[41] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming," In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (editors), *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press, pp. 123–131, 1996.

[42] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming," In *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware (ICES-96)*. Lecture Notes in Computer Science, vol. 1259. Berlin: Springer-Verlag. pp. 312-326, 1996.

[43] J. R. Koza, D. Andre, F. H Bennett III, and M. A. Keane, "Use of automatically defined functions and architecture-altering operations in automated circuit synthesis using genetic programming," In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (editors), *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press, pp. 132–140, 1996.

[44] F. H Bennett III, J. R. Koza, D. Andre, and M. A. Keane, "Evolution of a 60 Decibel op amp using genetic programming," In Higuchi, T. (editor). *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware (ICES-96)*. Lecture Notes in Computer Science, vol. 1259. Berlin: Springer-Verlag. pp. 455-469, 1996.

[45] T. Quarles, A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE 3 Version 3F5 User's Manual*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, March 1994.

[46] A. B. Williams, and F. J. Taylor, *Electronic Filter Design Handbook*, Third Edition, New York, NY: McGraw-Hill, 1995.

[47] J. R. Koza, F. H Bennett III, J. Lohn, F. Dunlap, D. Andre, and M. A. Keane, "Evolution of a tri-state frequency discriminator for the source identification problem using genetic programming," In P. P. Wang (editor) *Proceedings of Joint Conference of Information Sciences*, vol. I, pp. 95 – 99, 1997.

[48] J. R. Koza, F. H Bennett III, J. Lohn, F. Dunlap, D. Andre, and M. A. Keane, "Use of architecture-altering operations to dynamically adapt a three-way analog source identification circuit to accommodate a new source," In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors). 1997. *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13–16, 1997, Stanford University*. San Francisco, CA: Morgan Kaufmann, pp. 213 – 221, 1997.

[49] Koza, John R., Bennett III, Forrest H, Lohn, Jason, Dunlap, Frank, Andre, David, and Keane, Martin A. Automated synthesis of computational circuits using genetic programming. *Proceedings of the 1997 IEEE Conference on Evolutionary Computation,* Piscataway, NJ: IEEE Press, pp. 447–452, 1997.

[50] J. R. Koza, F. H Bennett III, M. A. Keane, and D. Andre, "Evolution of a time-optimal fly-to controller circuit using genetic programming," In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors),*Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13–16, 1997, Stanford University*, San Francisco, CA: Morgan Kaufmann, pp. 207 – 212, 1997.

[51] J. R. Koza, F. H Bennett III, D. Andre, and M. A. Keane, "Evolution using genetic programming of a low-distortion 96 Decibel operational amplifier. *Proceedings of the 1997 ACM Symposium on Applied Computing, San Jose, California, February 28 – March 2, 1997*. New York: Association for Computing Machinery. pp. 207 - 216, 1997.

[52] D. Andre and J. R. Koza, "Parallel genetic programming: A scalable implementation using the transputer architecture, In Angeline, P. J. and Kinnear, K. E. Jr. (editors), 1996, *Advances in Genetic Programming 2*, Cambridge, MA: MIT Press, 1996.

[53] J. C. Clements, "Minimum-time turn trajectories to fly-to points," *Optimal Control Applications and Methods*, vol. 11, pp. 39-50, 1990.

[54] T. Sterling, P. Messina, and P. H. Smith, *Enabling Technologies for Petaflops Computing*, Cambridge, MA: The MIT Press, 1995.
[55] R. M. Kielkowski, *Inside SPICE: Overcoming the Obstacles of Circuit Simulation*, New York, NY: McGraw-Hill, 1995.

BIOGRAPHIES

John R. Koza received a B.A. and M.S. in Computer Science and a M. A. in Mathematics from the University of Michigan. He received his Ph. D. in Computer Science from the University of Michigan in 1972. From 1973 to 1987, he was Chairman, Chief Executive Officer, and co-founder of Scientific Games Inc. of Atlanta. From 1987 to present, he is President of Third Millennium Venture Capital Limited of Los Altos Hills, California. He is currently consulting professor in the Computer Science Department and Symbolic Systems Program at Stanford University. Since 1988, he has taught courses in genetic algorithms and genetic programming in the School of Engineering. He has also taught a course on artificial life in the School of Engineering and currently co-teaches a course in computational molecular biology in the School of Medicine.

Forrest H Bennett III is visiting scholar in the Computer Science Department at Stanford University. He received his B. S. degree in Applied Mathematics at the University of Colorado in 1985. He ran a software consulting business for 5 years, where he designed systems, including industry's leading industrial drive shaft design system. He then became the Chief Engineer at Manco Systems where he designed and implemented the company's primary software product which is used for data collection in manufacturing environments. He has done research on using functional languages for programming parallel computers. His current research involves using genetic programming to solve problems and has published more than 24 papers in areas such as automatic programming of multi-agent systems, analog circuit design, and programming field programmable gate arrays.

David Andre is a Ph. D. candidate in the Computer Science Division of the University of California at Berkeley. He graduated with a B. S. in Symbolic Systems and a B. A. in Psychology from Stanford in 1994. He has been researching evolutionary computation (EC) for five years, and has published more than 25 papers. The focus of his research efforts has been to develop techniques that allow evolutionary computation to solve difficult, real-world problems. He is also the author of a public domain genetic programming kernel. His research spans several areas outside the EC field as well, including: autonomous robotics, vision, reinforcement learning, parallel processing, artificial intelligence and cognitive science. In 1996, he was awarded a National Defense Science and Engineering Grant for graduate research. He additionally works with Blue Pumpkin Software Inc. in designing scheduling software for call centers.

Martin A. Keane received his Ph. D. from Northwestern University in 1969. He had earlier received the B. S. E. E. from Illinois Institute of technology in 1961 and the M. S. Mathematics from the University of Hawaii in 1967. Between l972 and l976, Dr. Keane supervised the Operations Research group in the Mathematics Department at General Motors Research Laboratory in Warren, Michigan. He joined Bally Manufacturing Corporation of Chicago in l976 as Vice-President of Engineering for the amusement game division. He became Vice-President for Technology for the entire corporation in 1980, supervising over 200 engineers. In this capacity, after the sale of Scientific Games Inc. to Bally in 1982, Dr. Keane also designed and manufactured various products for Scientific Games, Inc., including video player-activated lottery game machines, interactive laser disk games, and on-line clerk-activated lottery game terminals with distributed fault-tolerant computer processing. He has also designed specialized equipment for the high-speed computer-controlled imaging operations used in printing instant lottery game tickets. Drs. Keane and Koza have received numerous patents in various areas of computers, electronics, printing, and gaming technology.

Frank Dunlap received the B. S. E. E. degree in electrical engineering from Mississippi State University in 1982, and the M. S. E. E. degree from the University of California at Berkeley in 1985. From 1985 to 1993, he was employed by Sierra Semiconductor Corporation where he worked, first as a designer, and later as a designer and manager in the area of mixed-signal CMOS integrated circuit design. In 1993, he became an integrated circuit design consultant. Recently, he co-founded Enabling Technology, Inc. in Palo Alto, California, a provider of high-performance analog integrated-circuit-level modules targeted for inclusion in mixed-signal integrated circuits.

Fig. 1 A one-input, one-output embryonic circuit used for the source identification circuit, the computational circuit, and the voltage reference circuit. The input appears at VSOURCE and the output is probed at VOUT. The circled wire Z0 is modifiable. RSOURCE and RLOAD are resistors. Nodes are numbered from 0 to 5.
Fig. 2a Modifiable wire `Z0` connecting nodes 1 and 2. The symbols C2 - C5 refer to capacitors.

Fig. 2b Result of applying the resistor-creating `R` function to modifiable wire Z0 of Fig. 2a, thereby transforming wire Z0 into a resistor R1.

Fig. 3 Result of applying transistor-creating `Q_POS_COLL_NPN` function to modifiable wire Z0 of Fig. 2a. The wire Z0 is replaced by the transistor Q6 connected to nodes 1 and 2 and the positive power supply POS.

Fig. 4 Result of applying transistor-creating `Q_3_NPN0` function to resistor R1 of Fig. 2b, thereby transforming it into transistor Q6.

Fig. 5 Result after applying the `SERIES` function to resistor R1 of Fig. 2b, thereby transforming it into resistors R7 and R1 and wire Z6.

Fig. 6 Result of applying the `PSS` parallel division function to resistor R1 of Fig. 2b, thereby transforming it into resistors R7 and R1 and wires Z6 and Z8.

Fig. 7 Result of applying the `T_GND_0` function to resistor R1 of Fig. 2b, thereby creating a connection to ground.

Fig. 8 One-input, two-output embryonic circuit for the crossover (woofer and tweeter) filter with a small portion of an illustrative circuit-constructing program tree. The two outputs are VOUT1 and VOUT2. The functions L, C, and C in the program tree refer to inductor- and capacitor-creating functions, respectively. See text for the application of these functions.
Fig. 9 Two-input, one-output embryonic circuit for the time-optimal controller. The two inputs are VSOURCE1 and VSOURCE2.

Fig. 10 Zero-input, one-output embryonic circuit for the temperature-sensing circuit. Z0 and Z1 are modifiable wires.

Fig. 11 One-input, one-output feedback embryo for an amplifier.

Fig. 12 One-input, one-output embryonic circuit for the lowpass filter.

Fig. 13 Detailed flowchart for calculation of circuit fitness.

Fig. 14a Genetically evolved ladder lowpass filter with seven series inductors (located horizontally across the top) and seven shunt capacitors (located vertically).

Fig. 14b Frequency domain behavior of genetically evolved ladder lowpass filter. The filter delivers about 1V for frequencies up to 1 kHz and almost 0 V for frequencies greater than 2 kHz.

Fig. 15a Genetically evolved "bridged T" lowpass filter. The "T" consists of capacitors C3 and C15 and inductor L11 and the "bridge" consists of inductor L14.

Fig. 15b Genetically evolved elliptic (Cauer) lowpass filter with inductors located horizontally across the top of the figure and five vertical shunts, each containing one inductor and one capacitor in series.

Fig. 16a Genetically evolved crossover (woofer and tweeter) filter.

Fig. 16b Frequency domain behavior of genetically evolved crossover filter.

Fig. 17a Genetically evolved four-way source identification circuit.

Fig. 17b Frequency domain behavior of genetically evolved four-way source identification circuit. The three boxes indicate the range of allowable output voltages (along the linear vertical axis) for the specified range of input frequencies (along the logarithmic horizontal axis).
Fig. 18a Genetically evolved cube root circuit.
Fig. 18b Performance of the genetically evolved cube root circuit. The actual output and the target output voltages (along the linear vertical axis) are overlain and virtually identical.
Fig. 19 Best genetically evolved time-optimal controller.
Fig. 20a Best genetically evolved amplifier.

Fig. 20b `ADF0` for best genetically evolved amplifier.

Fig. 21a Best-of-run temperature-sensing circuit.

Fig. 21b Performance of the genetically evolved temperature-sensing circuit (with output voltage along the linear vertical axis).

Fig. 22a Best voltage reference circuit from generation 0.
Fig. 22b Close-up of output of best voltage reference circuit from generation 0.

Fig. 23 Output (with voltage along the linear vertical axis) of the best voltage reference circuits from generations 0, 6, 49, and 80 for temperatures of 0°, 25°, 50°, 75°, and 100° C.

Fig. 24 Best voltage reference circuit from generation 6.
Fig. 25 Best voltage reference circuit from generation 49.
Fig. 26 Best voltage reference circuit from generation 80.

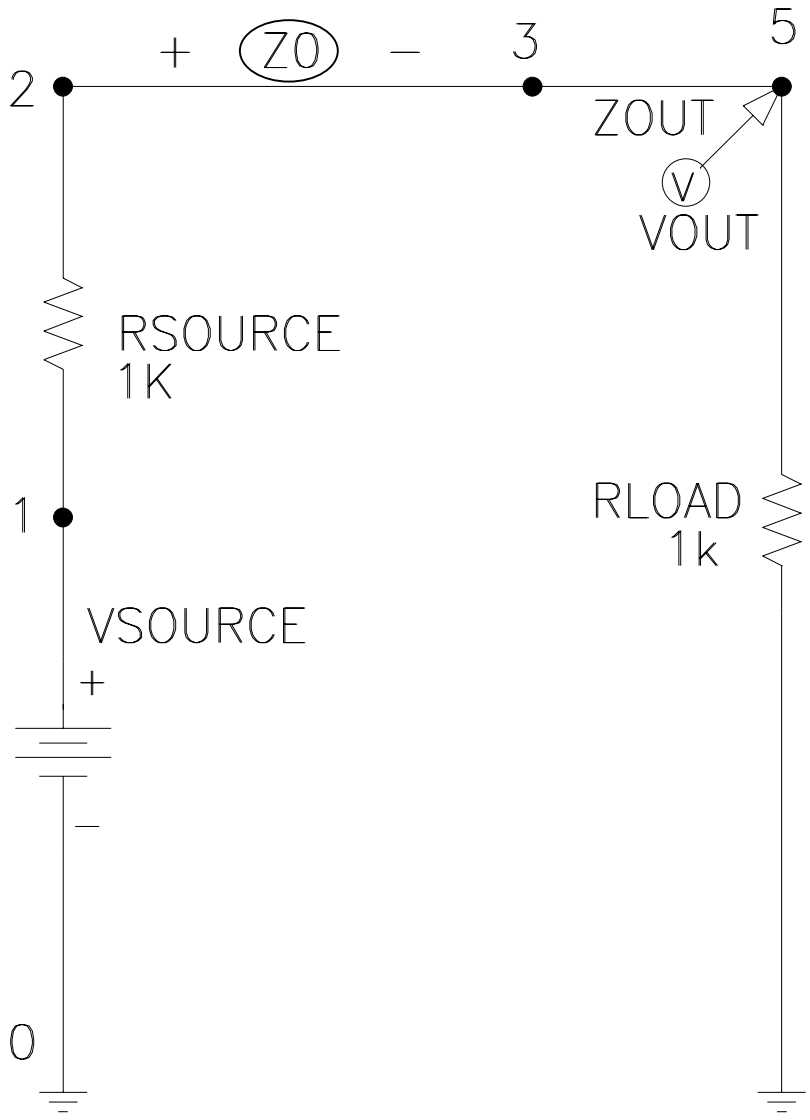

Fig. 1 A one-input, one-output embryonic circuit used for the source identification circuit, the computational circuit, and the voltage reference circuit. The input appears at VSOURCE and the output is probed at VOUT. The circled wire Z0 is modifiable. RSOURCE and RLOAD are resistors. Nodes are numbered from 0 to 5.

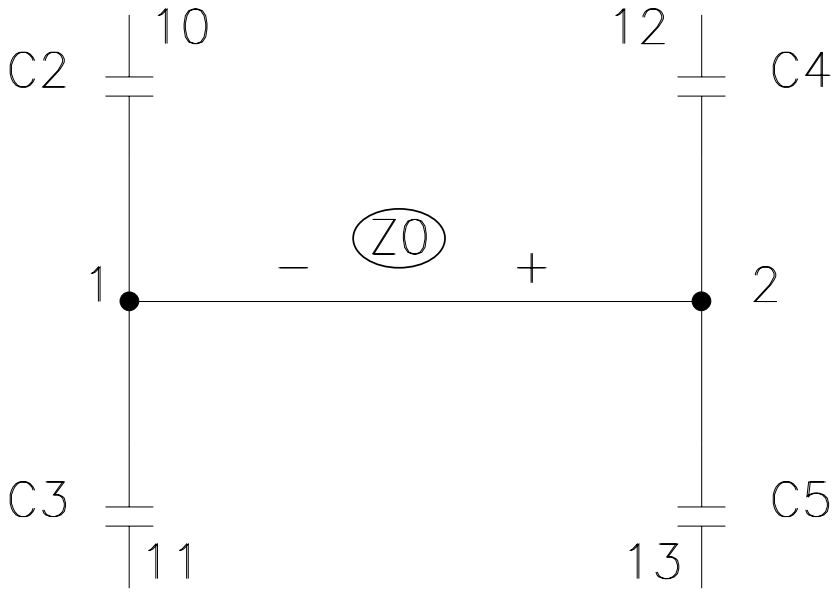

Fig. 2a Modifiable wire Z0 connecting nodes 1 and 2. The symbols C2 - C5 refer to capacitors.

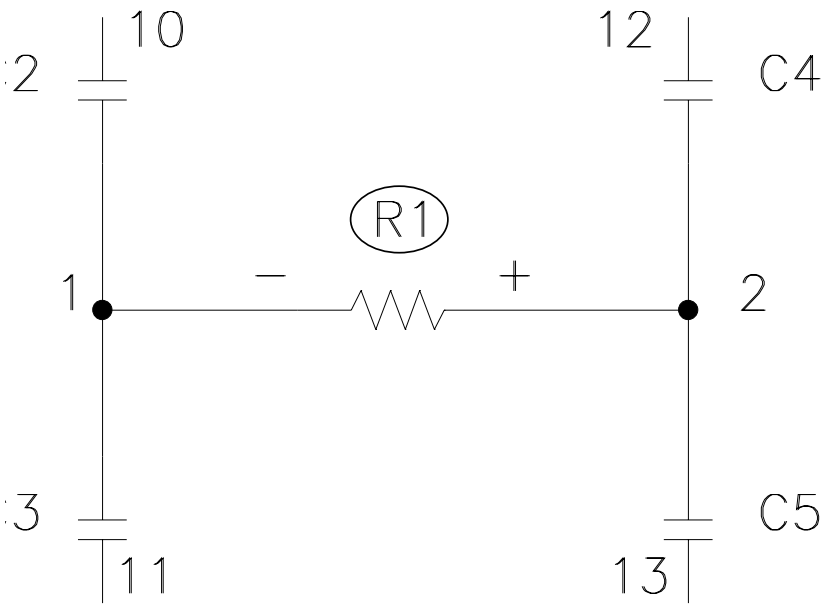

Fig. 2b Result of applying the resistor-creating R function to modifiable wire Z0 of Fig. 2a, thereby transforming wire Z0 into a resistor R1.

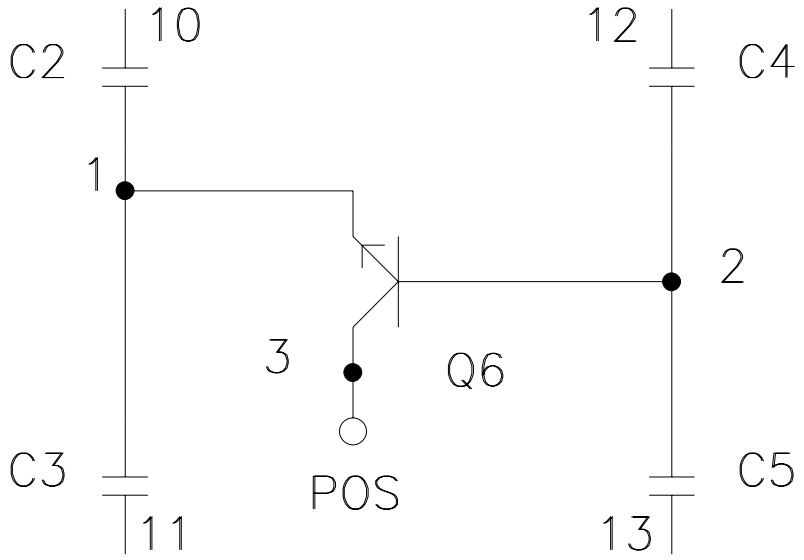

Fig. 3 Result of applying transistor-creating `Q_POS_COLL_NPN` function to modifiable wire Z0 of Fig. 2a. The wire Z0 is replaced by the transistor Q6 connected to nodes 1 and 2 and the positive power supply POS.

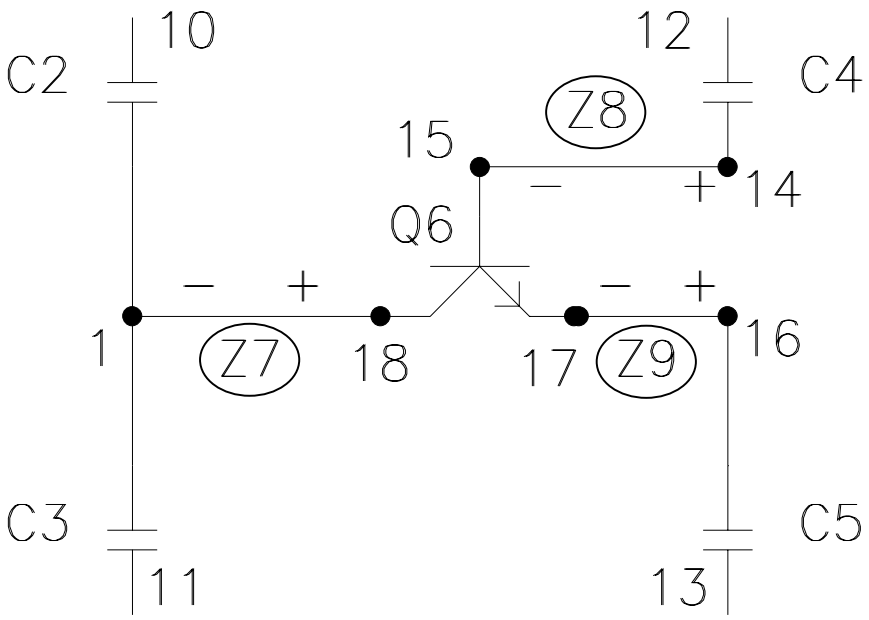

Fig. 4 Result of applying transistor-creating `Q_3_NPN0` function to resistor R1 of Fig. 2b, thereby transforming it into transistor Q6.

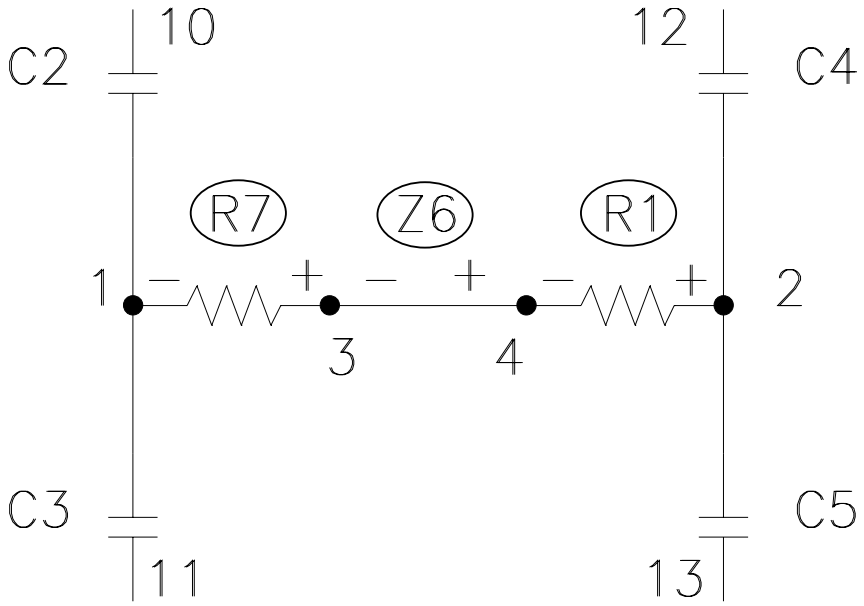

Fig. 5 Result after applying the `SERIES` function to resistor R1 of Fig. 2b, thereby transforming it into resistors R7 and R1 and wire Z6.

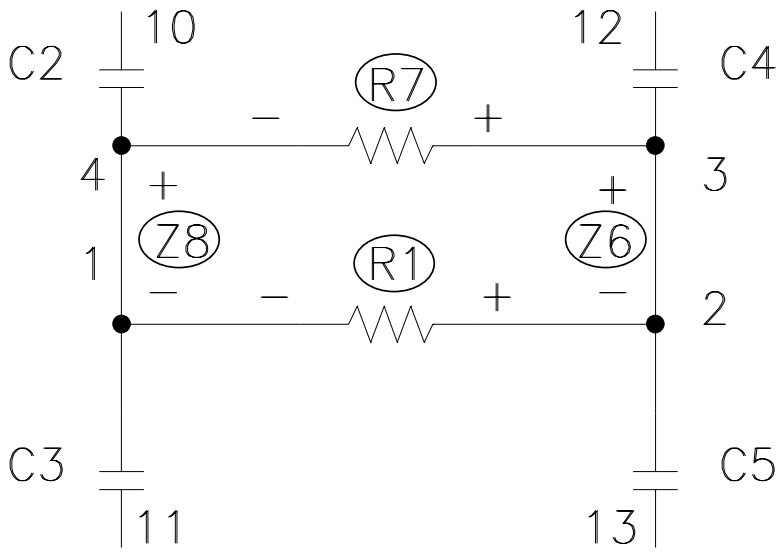

Fig. 6 Result of applying the PSS parallel division function to resistor R1 of Fig. 2b, thereby transforming it into resistors R7 and R1 and wires Z6 and Z8.

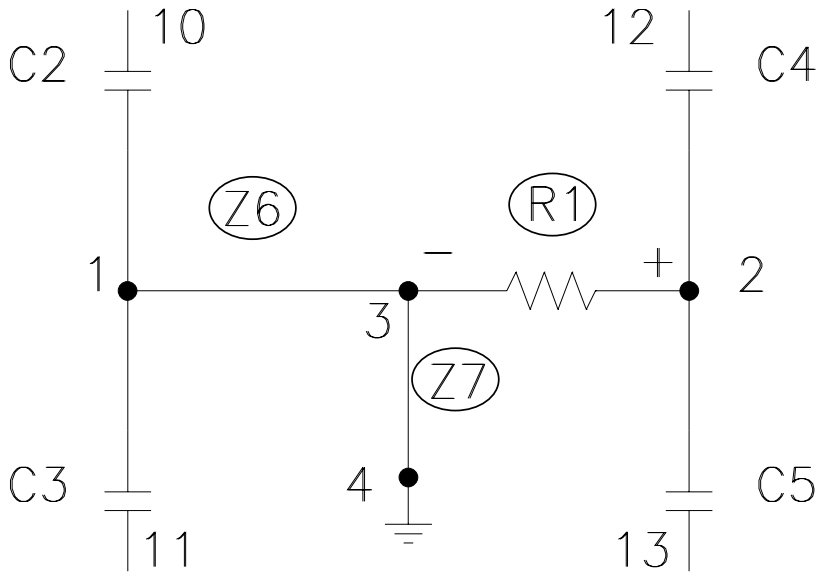

Fig. 7 Result of applying the `T_GND_0` function to resistor R1 of Fig. 2b, thereby creating a connection to ground.

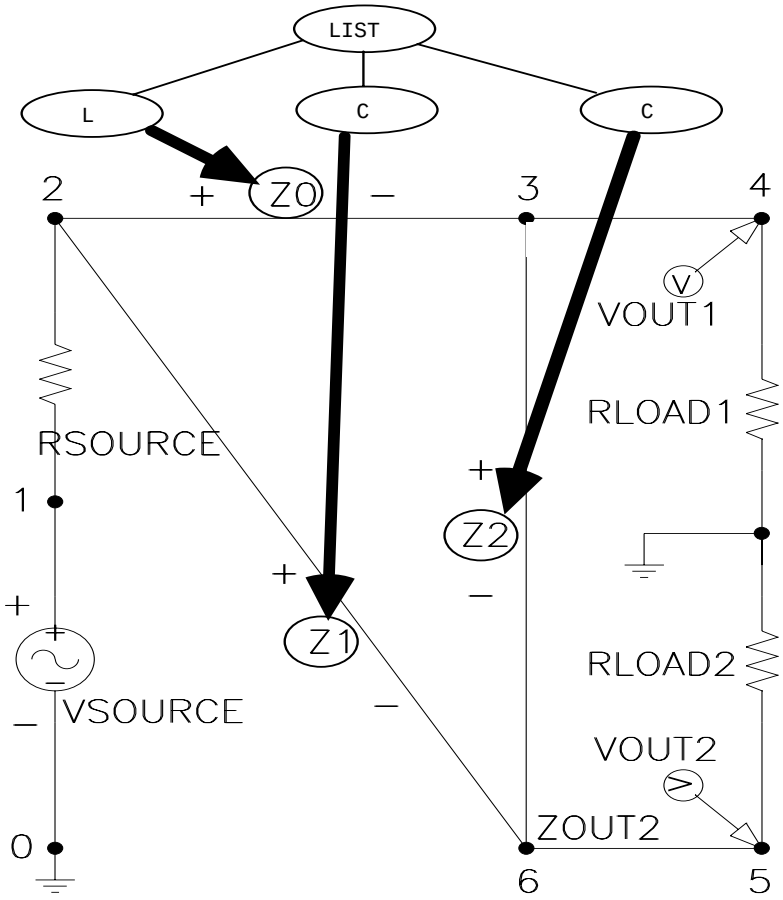

Fig. 8 One-input, two-output embryonic circuit for the crossover (woofer and tweeter) filter with a small portion of an illustrative circuit-constructing program tree. The two outputs are VOUT1 and VOUT2. The functions L, C, and C in the program tree refer to inductor- and capacitor-creating functions, respectively. See text for the application of these functions.

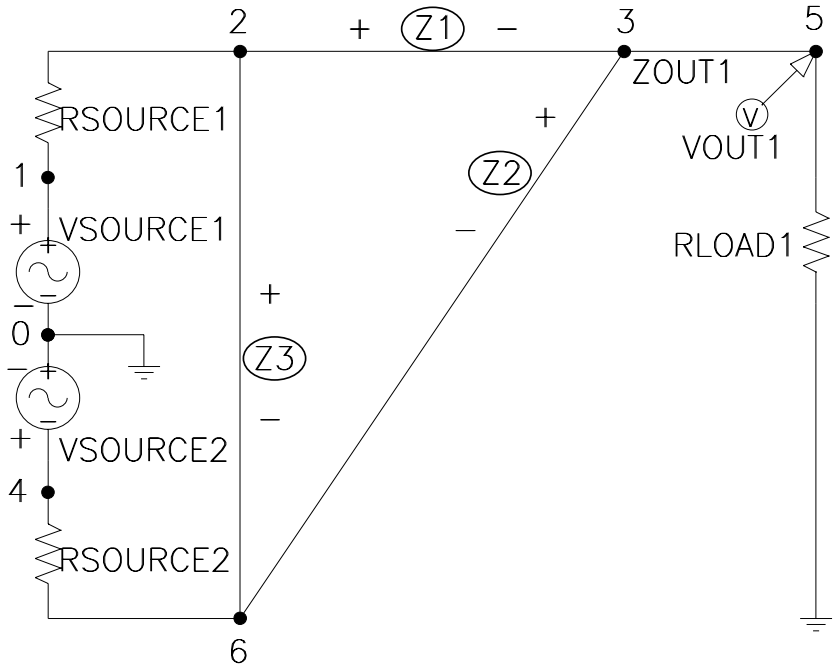

Fig. 9 Two-input, one-output embryonic circuit for the time-optimal controller. The two inputs are VSOURCE1 and VSOURCE2.

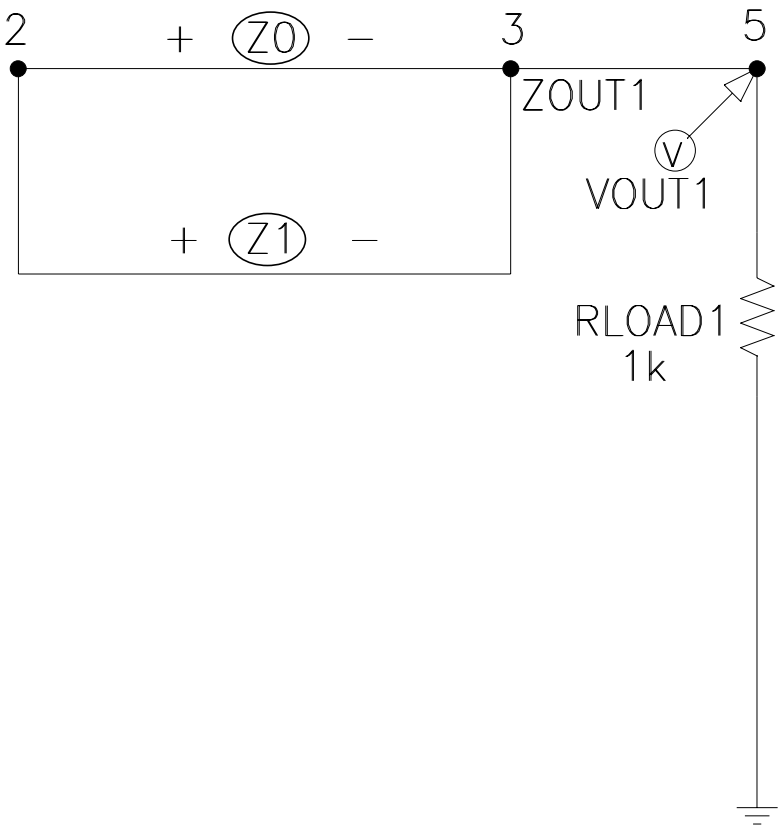

Fig. 10 Zero-input, one-output embryonic circuit for the temperature-sensing circuit. Z0 and Z1 are modifiable wires.

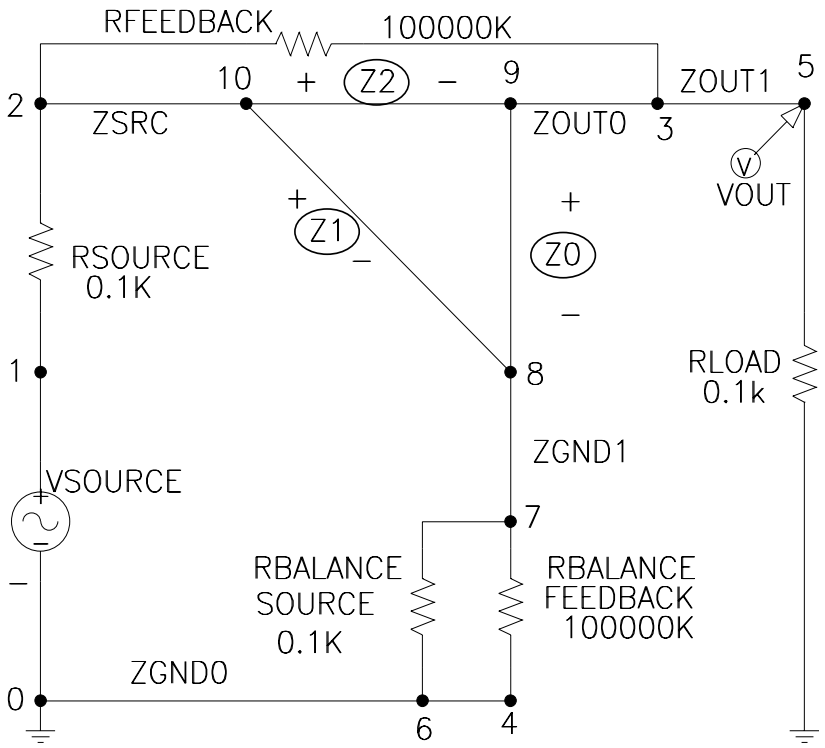

Fig. 11 One-input, one-output feedback embryo for an amplifier.

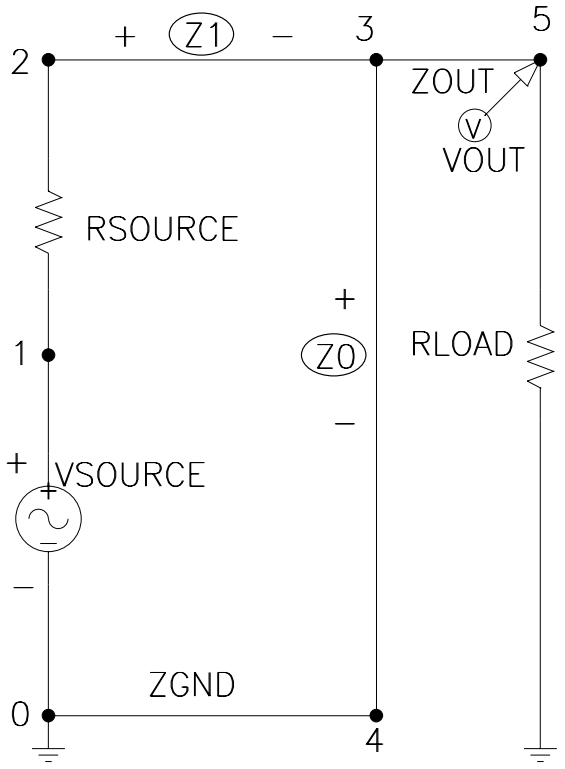

Fig. 12 One-input, one-output embryonic circuit for the lowpass filter.

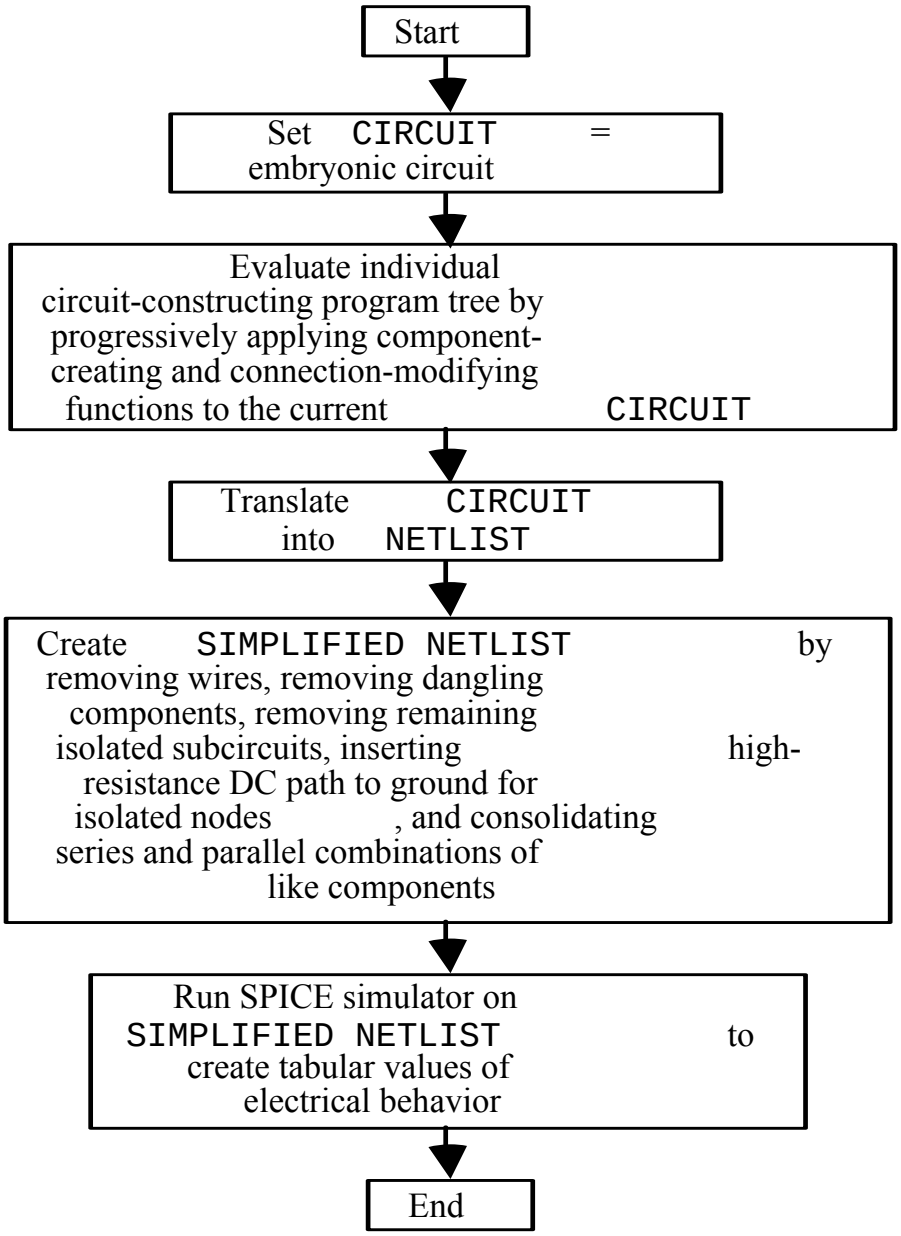

Fig. 13 Detailed flowchart for calculation of circuit fitness.

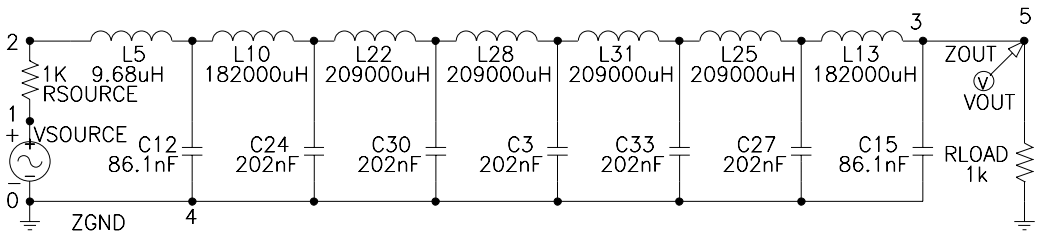

Fig. 14a Genetically evolved ladder lowpass filter with seven series inductors (located horizontally across the top) and seven shunt capacitors (located vertically).

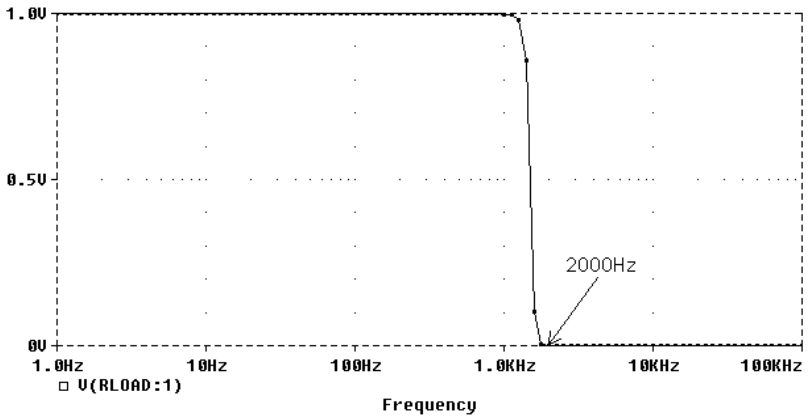

Fig. 14b Frequency domain behavior of genetically evolved ladder lowpass filter. The filter delivers about 1V for frequencies up to 1 kHz and almost 0 V for frequencies greater than 2 kHz.

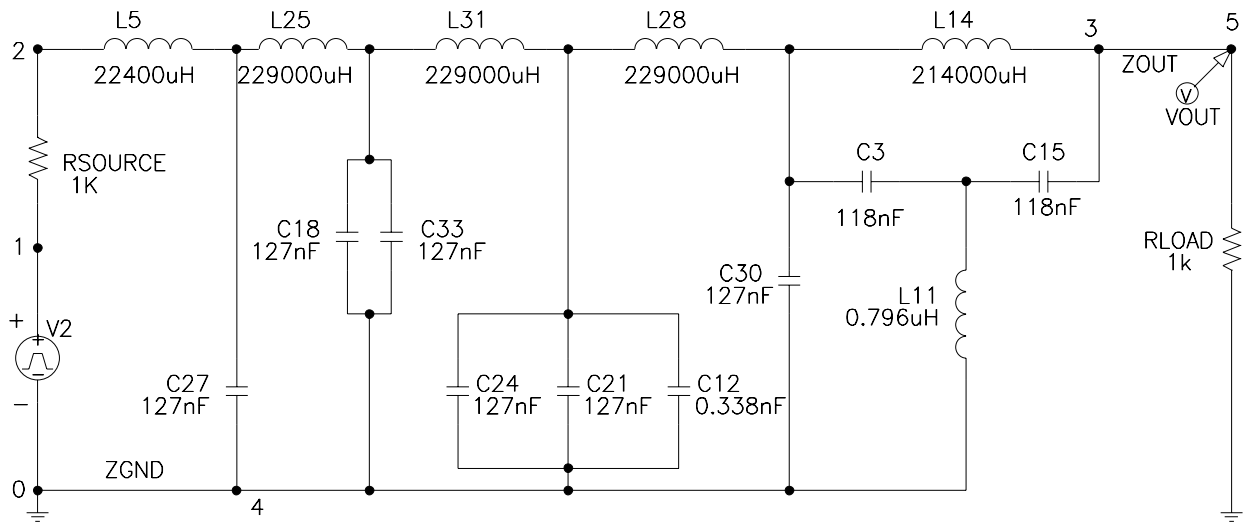

Fig. 15a Genetically evolved "bridged T" lowpass filter. The "T" consists of capacitors C3 and C15 and inductor L11 and the "bridge" consists of inductor L14.

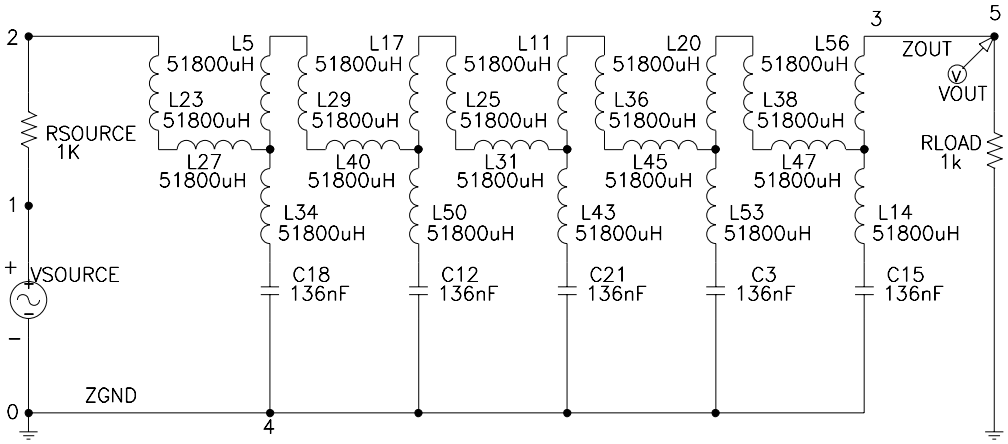

Fig. 15b Genetically evolved elliptic (Cauer) lowpass filter with inductors located horizontally across the top of the figure and five vertical shunts, each containing one inductor and one capacitor in series.

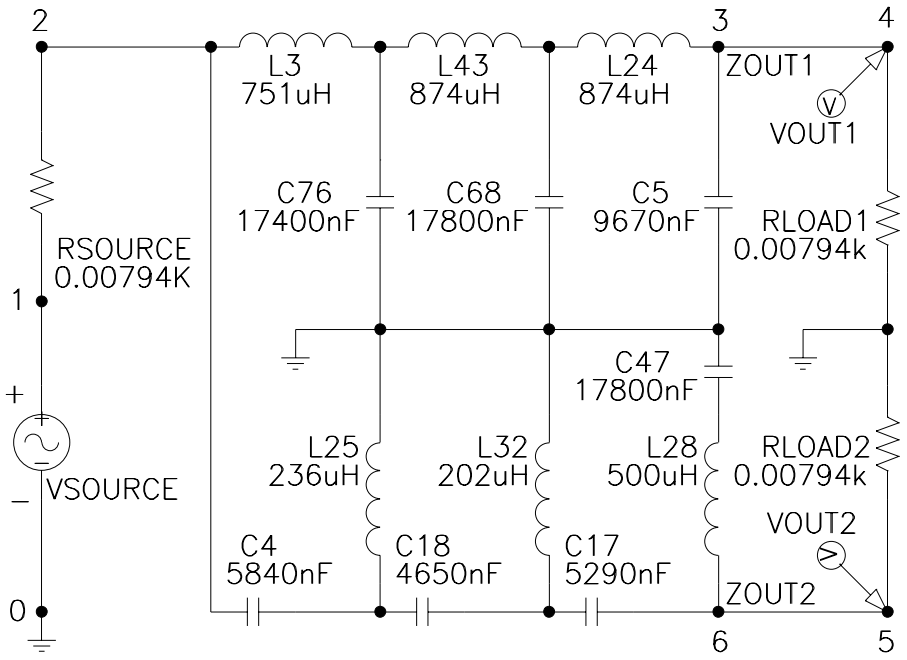

Fig. 16a Genetically evolved crossover (woofer and tweeter) filter.

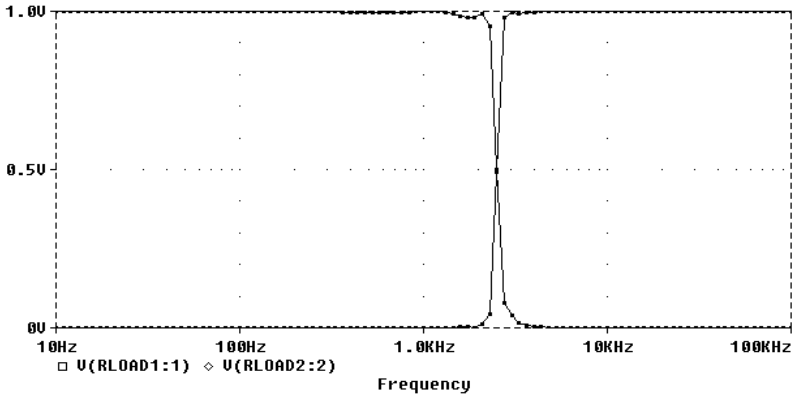

Fig. 16b Frequency domain behavior of genetically evolved crossover filter.

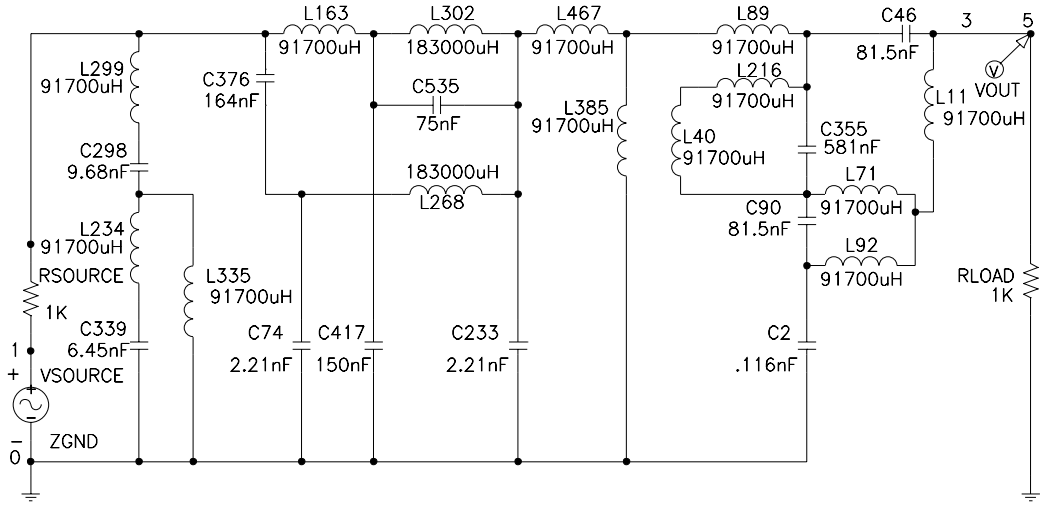

Fig. 17a Genetically evolved four-way source identification circuit.

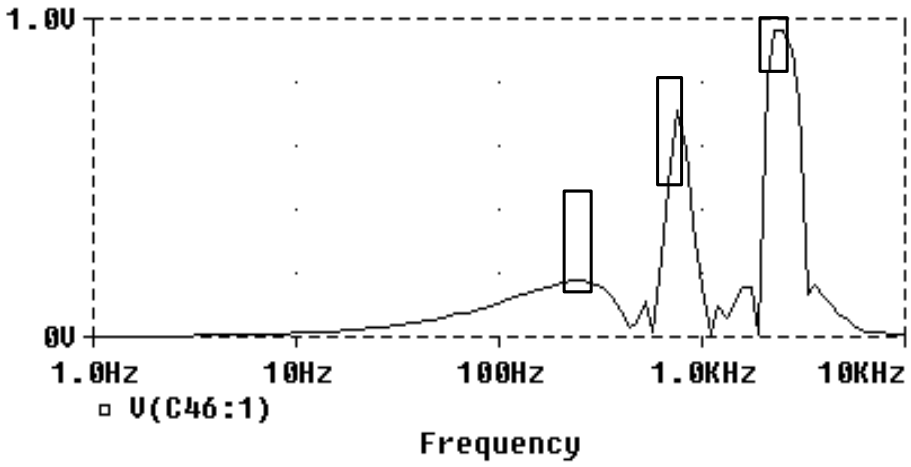

Fig. 17b Frequency domain behavior of genetically evolved four-way source identification circuit. The three boxes indicate the range of allowable output voltages (along the linear vertical axis) for the specified range of input frequencies (along the logarithmic horizontal axis).

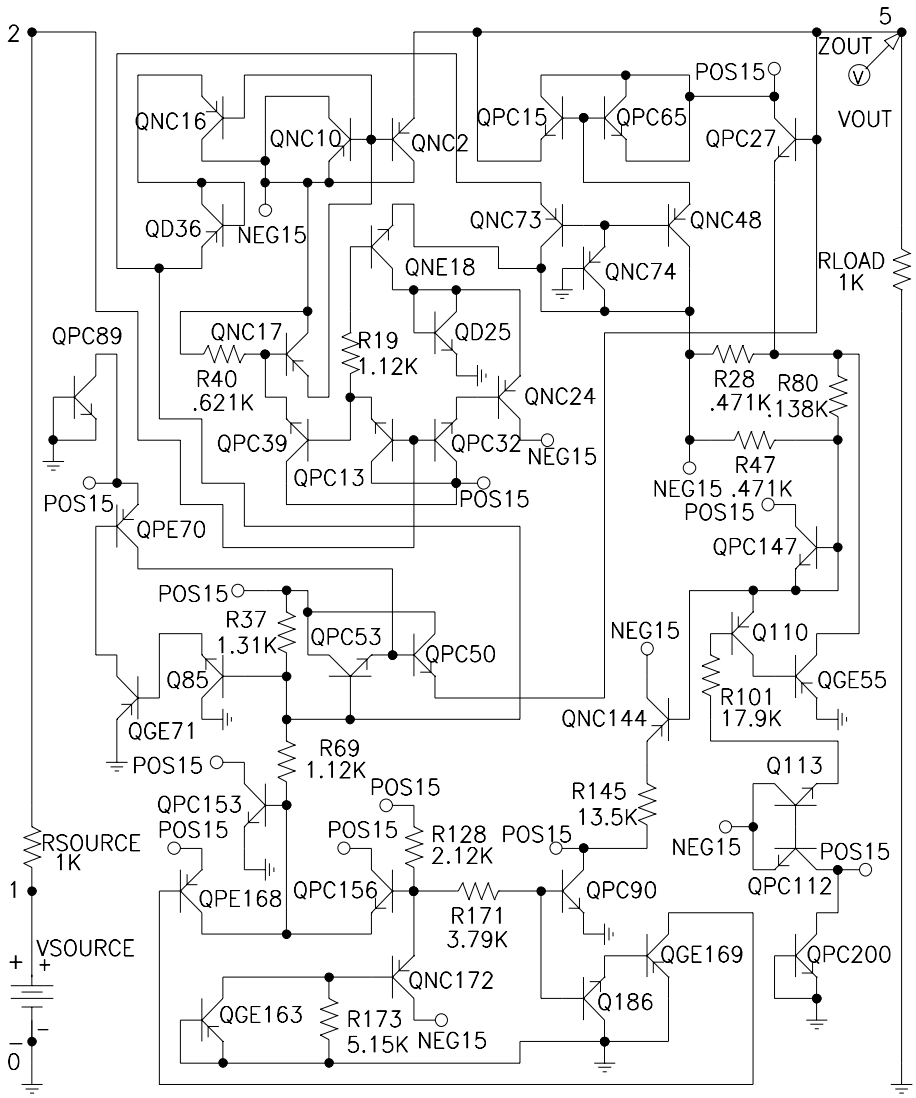

Fig. 18a Genetically evolved cube root circuit.

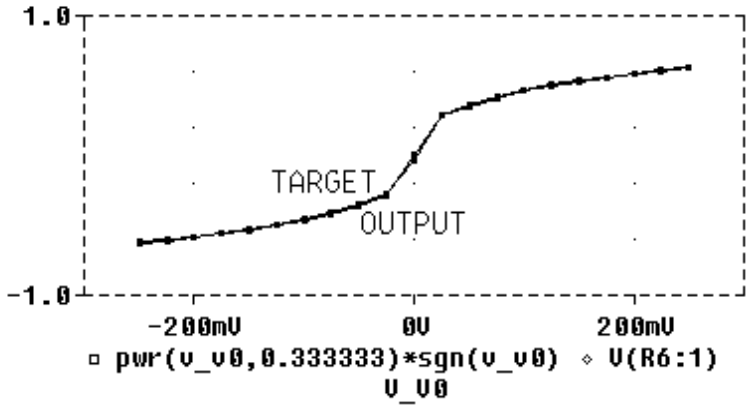

Fig. 18b Performance of the genetically evolved cube root circuit. The actual output and the target output voltages (along the linear vertical axis) are overlain and virtually identical.

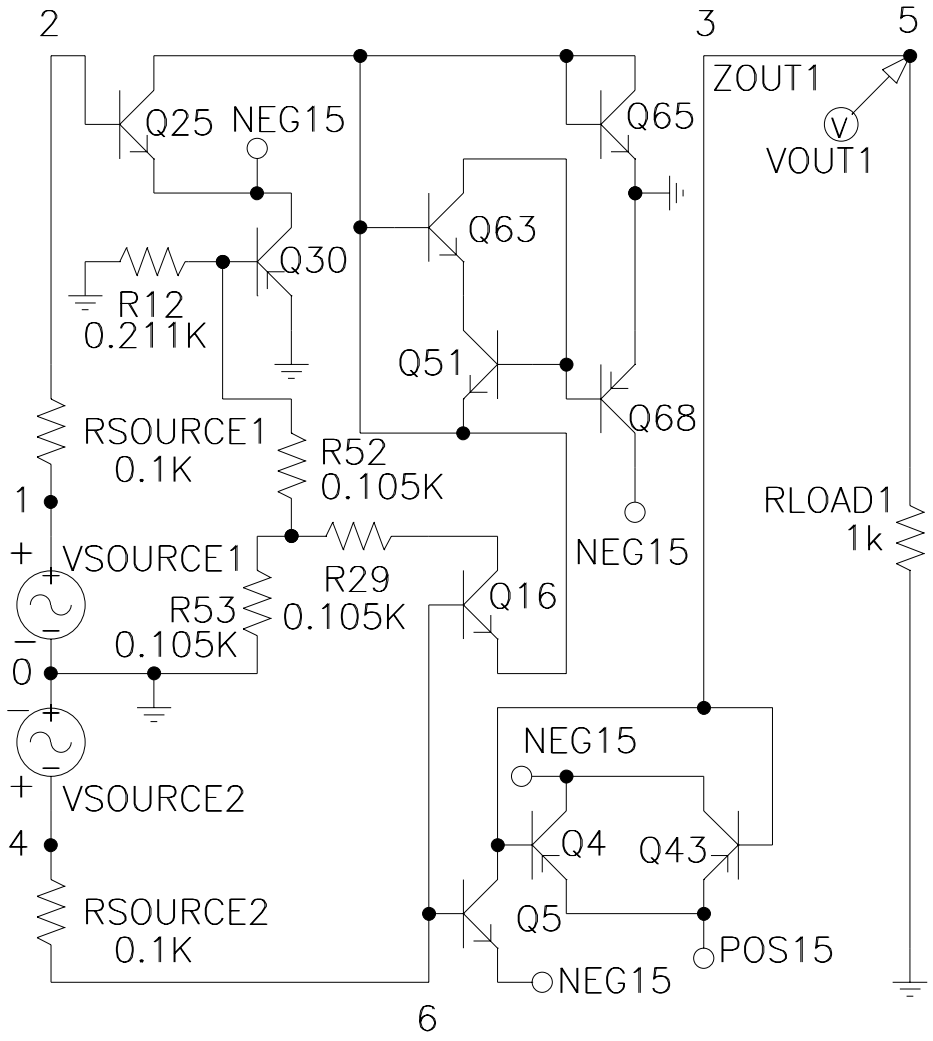

Fig. 19 Best genetically evolved time-optimal controller.

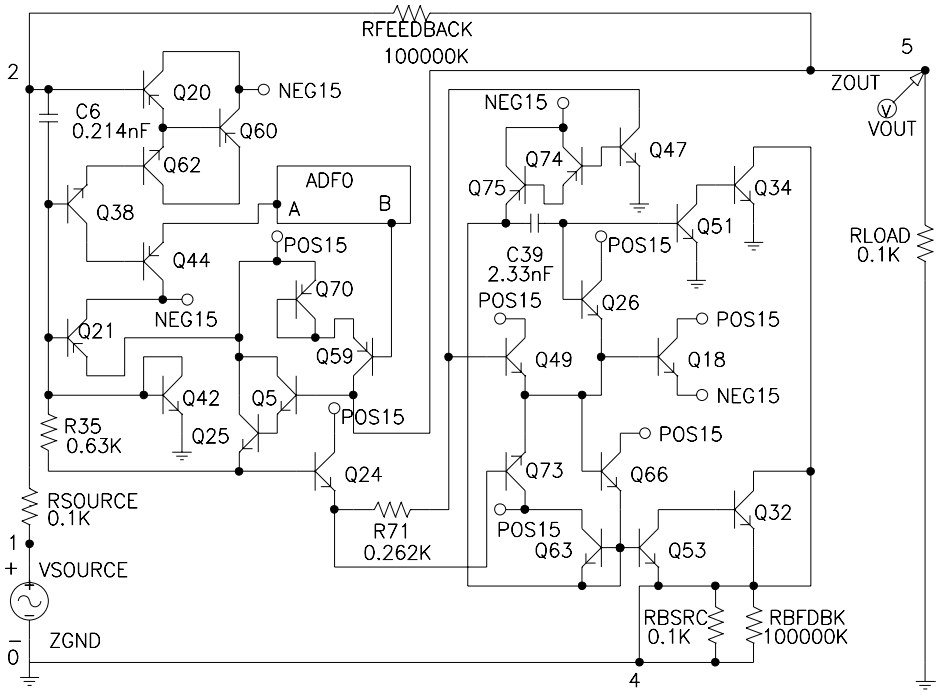

Fig. 20a Best genetically evolved amplifier.

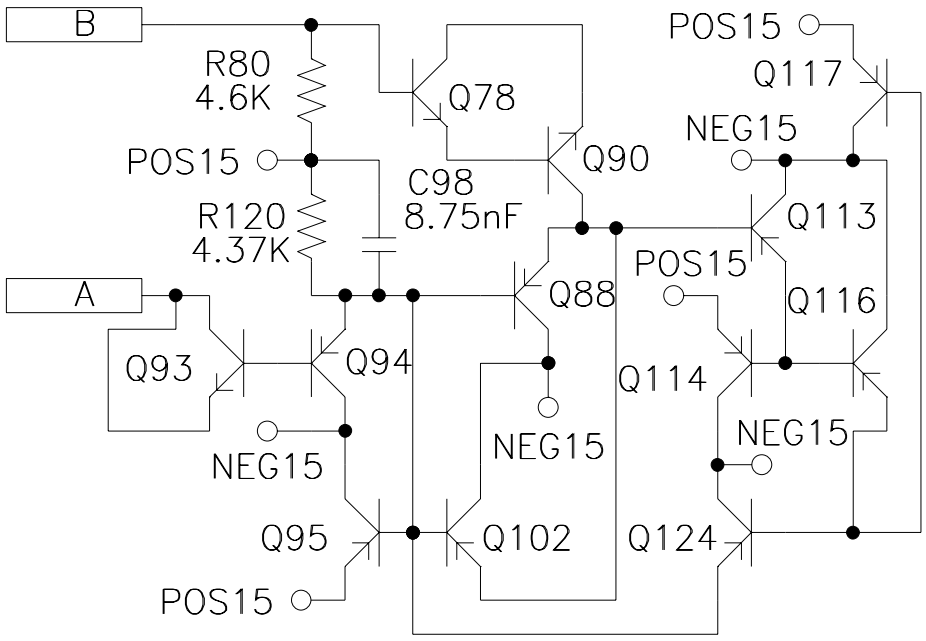

Fig. 20b `ADF0` for best genetically evolved amplifier.

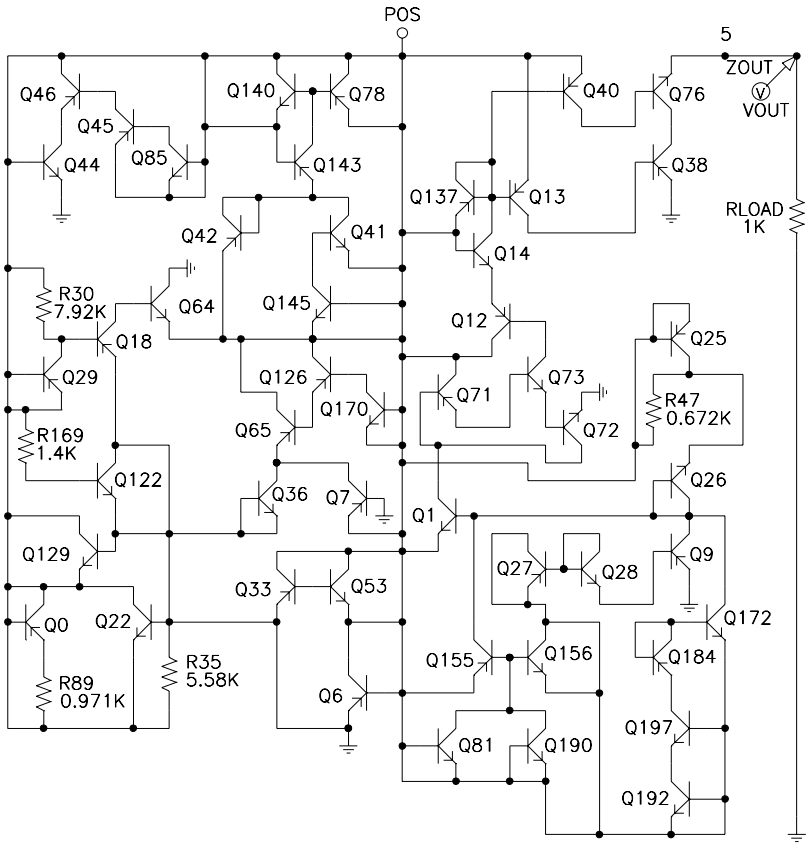

Fig. 21a Best-of-run temperature-sensing circuit.

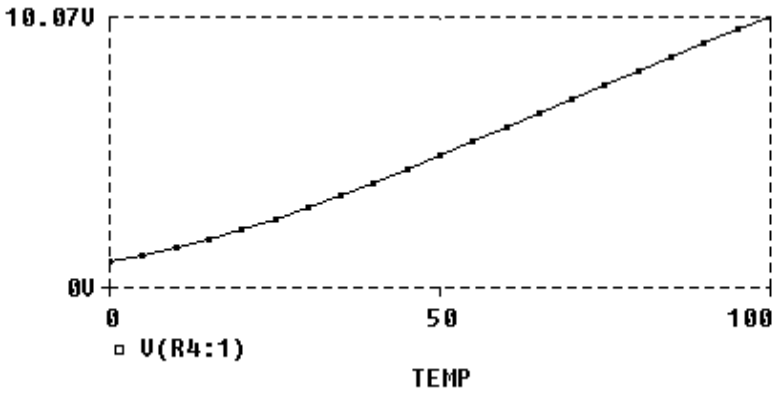

Fig. 21b Performance of the genetically evolved temperature-sensing circuit (with output voltage along the linear vertical axis).

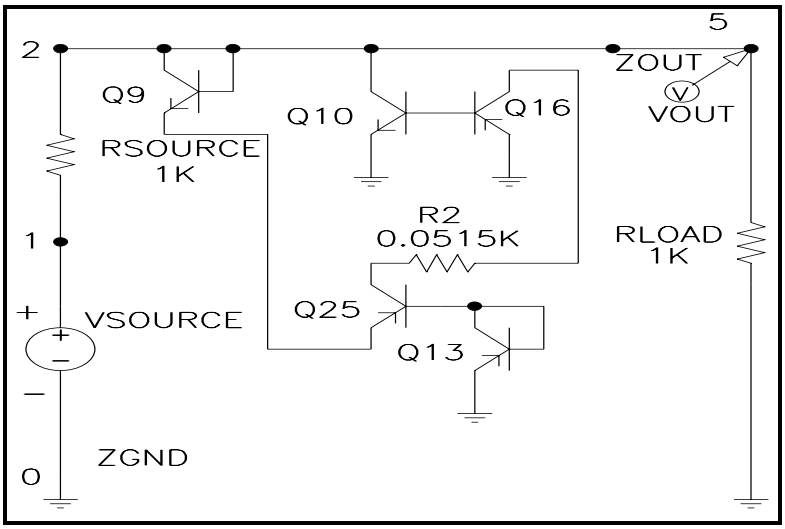

Fig. 22a Best voltage reference circuit from generation 0.

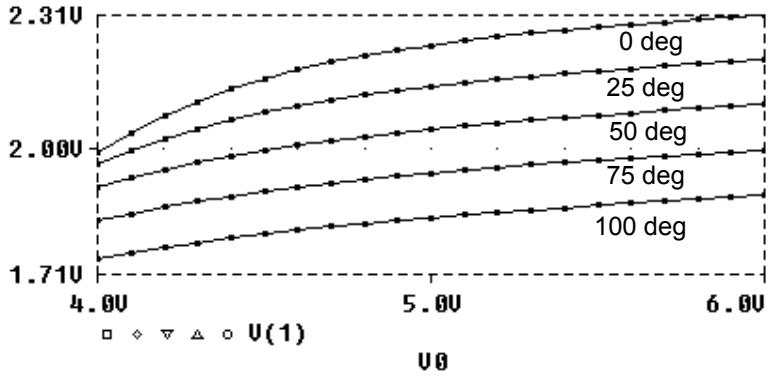

Fig. 22b Close-up of output of best voltage reference circuit from generation 0.

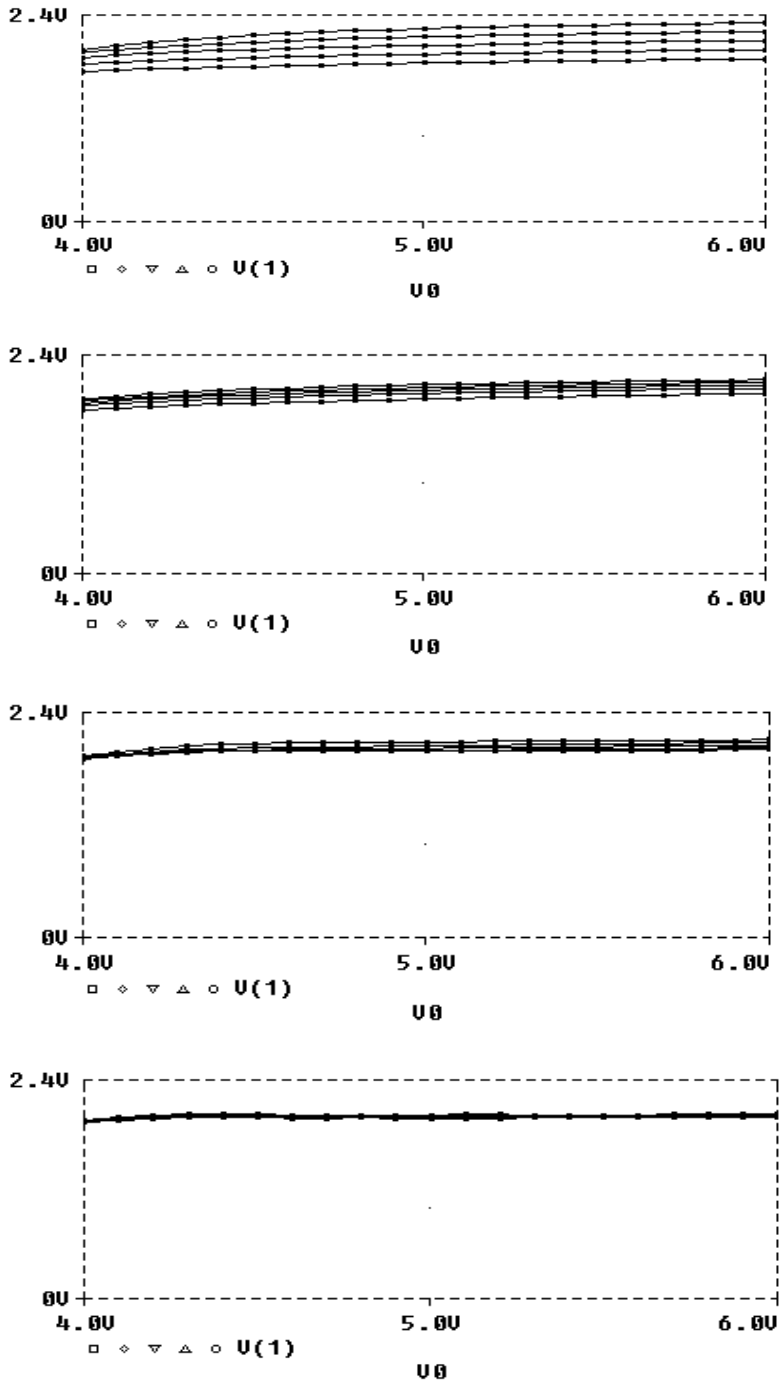

Fig. 23 Output (with voltage along the linear vertical axis) of the best voltage reference circuits from generations 0, 6, 49, and 80 for temperatures of 0°, 25°, 50°, 75°, and 100° C.

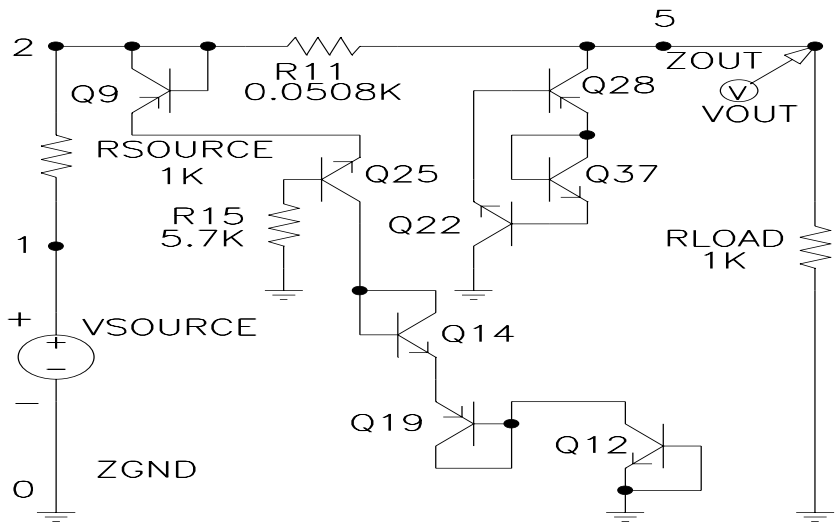

Fig. 24 Best voltage reference circuit from generation 6.

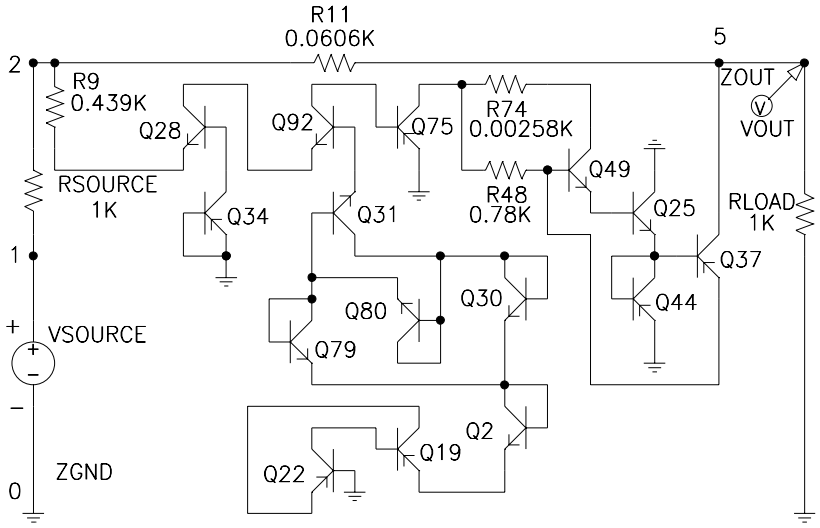

Fig. 25 Best voltage reference circuit from generation 49.

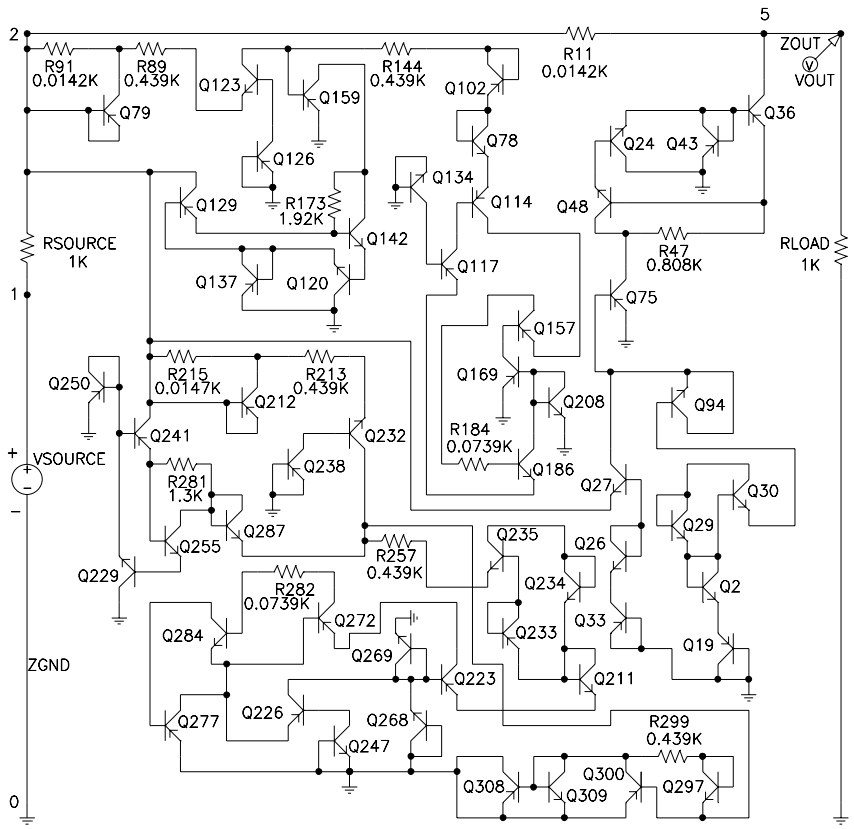

Fig. 26 Best voltage reference circuit from generation 80.

Paper TEC #26 – Version 4 - Submitted June 12, 1997 to *IEEE Transactions on Evolutionary Computation*.

Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming

John R. Koza, *Member*, *IEEE*, Forrest H Bennett III, *Member*, *IEEE*, David Andre, Martin A. Keane, *Member*, *IEEE*, Frank Dunlap, *Member*, *IEEE*

NOTE TO TYPESETTER: The middle name of Forrest H Bennett III is the single letter "H" WITH NO PERIOD (like "Harry S Truman"). It appears repeatedly herein in author information and bibliography.