AUTOMATIC SYNTHESIS OF ELECTRICAL CIRCUITS USING DEVELOPMENTAL GENETIC PROGRAMMING

PART 1 — METHODS

OUTLINE

- **The design process**
- **Circuit Synthesis using developmental genetic programming**
  - Initial circuit (embryo plus test fixture)
  - Circuit-constructing program trees
    - Component-creating functions
    - Topology-modifying functions
    - Development-controlling functions

EXAMPLES

- **Filters (lowpass, highpass, bandpass bandstop)**
- **Filters (Campbell, Zobel, Johnson, Butterworth, Chebychev, Cauer [elliptic])**
- **Crossover filters (woofer-tweeter, woofer-midrange-tweeter)**
- **Source Identification Problem (three-way and four-way — with changing environment)**
- **Amplifiers (10 dB, 40 dB, 60 dB, 96 dB)**
- **Computational circuits (squaring, cubing, square root, cube root, logarithmic, Gaussian)**
- **Circuit for time-optimal fly-to controller**
- **Temperature-sensing circuit**
- **Voltage reference circuit**
- **Philbrick circuit**
- **NAND gate**
- **Digital-to-analog converter (DAC)**
- **6 post-2000 patented inventions**

DESIGN

- **Find a structure, composed of components of various types, that satisfies user-specified goals**

- **Design is a major activity of practicing engineers**

- **The design process typically entails tradeoffs between competing considerations**

- **The end product of the design process is usually a satisfactory and compliant design as opposed to a perfect design**

- **Design is usually viewed as requiring human intelligence**

DESIGN USING GENETIC PROGRAMMING

- **Can be done with a minimum of domain-specific knowledge and no mathematical analysis, abstractions, or formal models**
- **Starts with a set of ingredients and information concerning the number of inputs and outputs**
- **Is driven by the user's design goals (implemented as a single scalar "fitness" value).**
- **It's (almost) WYWIWYG – "What You Want Is What You Get" (pronounced "wow-eee-wig")**
- **However, as they, be careful what you ask for because the process is, more precisely, WYGIWYAF – "What You Get Is What You Ask For"**

THE SEARCH SPACE FOR EVEN A SIMPLE ELECTRICAL CIRCUIT IS VERY LARGE

ASSUMPTIONS FOR ILLUSTRATIVE EXAMPLE

- **Assume only a one-input, one-output circuit**
- **Assume we know exact size of ultimate circuit in advance and that it is only 20 components (Note: the number of components is, in general, unknowable in advance and 20 is small)**
- **Assume only two-leaded components**
- **Assume only 3 types of such components: Resistor (R), Capacitor (C), Inductor (I)**
- **Assume only 20 component values per decade (i.e., dividing each decade into 5% slices) and 10 decades of component values (i.e., a total of only 200 possible values)**

LARGE SEARCH SPACE FOR EVEN A SIMPLE ELECTRICAL CIRCUIT - CONTINUED

- **The calculation below is overly simplified, but suggestive of large size of the search space**
- **Then, given the above assumptions, there are 861 ways of choosing two leads from 42 leads (20 x 2 + 2 = 42) and thus a total of 2^{861} (~ 10^{260}) ways of making undirected connections between two leads**
- **There are 3^{20} (~ 10^{9}) possible ways of picking the type of component**
- **There are 200^{20} (~ 10^{52}) possible ways of the value of each of the 20 components**
- **Thus, there are approximately <u>up to</u> 10^{321} possible circuits with exactly 20 components (and with the above assumptions)**
- **And, of course, one would rarely know, in advance, that the requisite circuit had precisely 20 components**

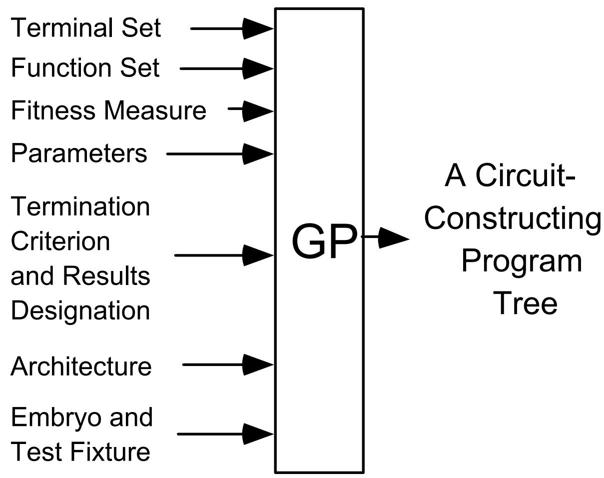
DIFFICULTY OF AUTOMATED ANALOG CIRCUIT DESIGN

- ***Analysis*** **of a circuit is itself difficult, and there is no previous general way to automate the *synthesis* of an analog electrical circuit from a high-level statement of the circuit's behavior and characteristics**
- **A hard problem**
  - Exponential in the number of components
  - More than 10^{300} circuits with a mere 20 components

DIFFICULTY OF AUTOMATED ANALOG CIRCUIT DESIGN — CONTINUED

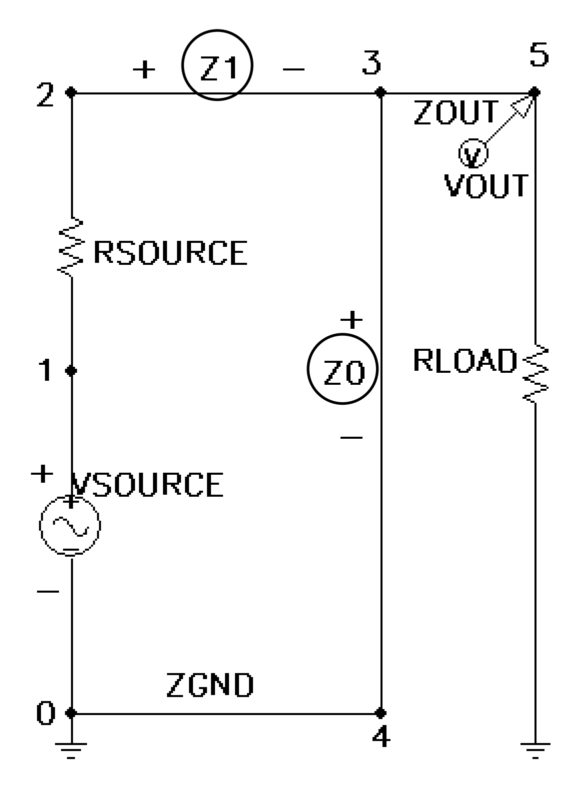
- **An important problem**
  - Too few analog designers
  - There is a comparatively small "egg shell" of analog circuitry around almost all digital circuits
  - The time required for the total design is often controlled by the time required to design the analog portion
  - Analog circuits must be redesigned with each new generation of solid-state process technology
  - Solid-state process technology is optimized for digital circuits – thus making analog design even more difficult

DEVELOPMENTAL GENETIC PROGRAMMING FOR AUTOMATED SYNTHESIS OF ANALOG CIRCUITS



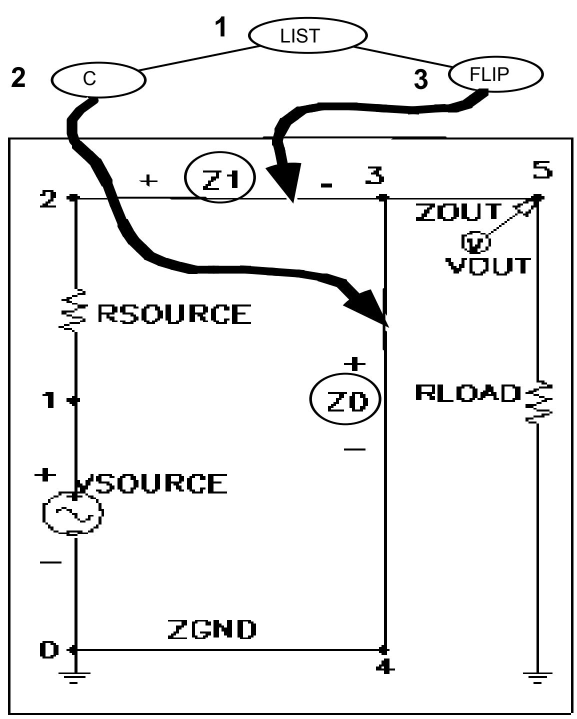
ONE-INPUT, ONE-OUTPUT INITIAL CIRCUIT

- **Initial circuit consists of embryo and test fixture**
- **Embryo has modifiable wires (e.g., Z0 AND Z1)**
- **Test fixture has input and output ports and usually has source resistor and load resistor. There are no modifiable wires (or modifiable components) in the test fixture.**



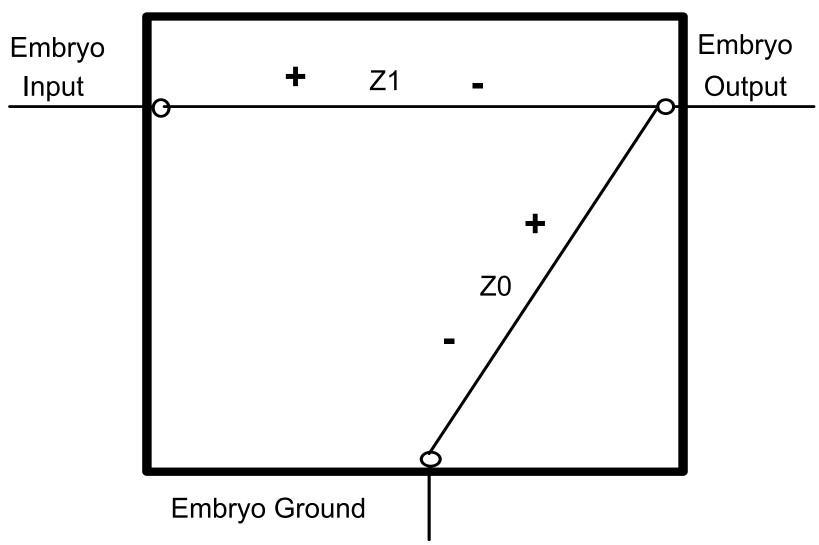
THE INITIAL CIRCUIT

- **Circuit-constructing program tree contains**
  - Component-creating functions
  - Topology-modifying functions
  - Development-controlling functions
- **Circuit-constructing program tree has one result-producing branch for each modifiable wire in embryo of the initial circuit**
- **There is a writing head linking each modifiable wire (or modifiable component) with one point of the program tree**



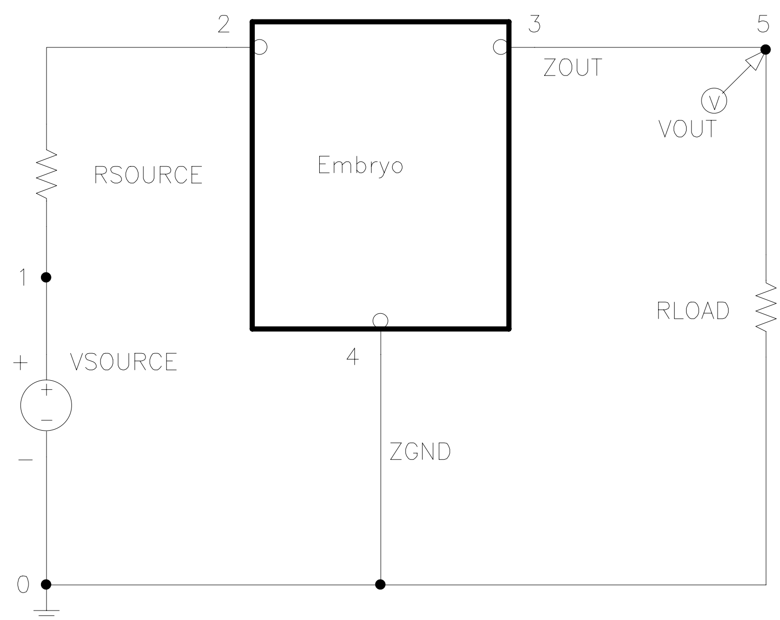
TWO PARTS OF AN INITIAL CIRCUIT

EMBRYO WITH TWO MODIFIABLE WIRES, Z0 AND Z1 AND THREE PORTS (EMBRYO_INPUT, EMBRYO_OUTPUT, AND EMBRYO_GROUND)



TWO PARTS OF AN INITIAL CIRCUIT

ONE-INPUT, ONE-OUTPUT TEST FIXTURE WITH THREE PORTS TO THE EMBRYO



TWO PARTS OF AN INITIAL CIRCUIT

THE EMBRYO

- **An embryo contains at least one modifiable wire. A modifiable wire is a wire that is capable of being converted into electrical components, other modifiable wires, and nonmodifiable wires during the developmental process.**
- **The embryo is very simple - often as little as one (or a few) modifiable wires**
- **An embryo has one or more ports that enable it to be embedded into a test fixture.**
- **When the embryo is embedded in the test fixture, the result is typically a useless and degenerate circuit**

TWO PARTS OF AN INITIAL CIRCUIT

THE EMBRYO - CONTINUED

- **Occasionally, an embryo may also contain non-modifiable wires, non-modifiable electrical components, or modifiable electrical components.**
- **The developmental process operates on the embryo (not the test fixture)**
- **The distinctive feature of the embryo is that it contains at least one modifiable wire.**

TWO PARTS OF AN INITIAL CIRCUIT

THE TEST FIXTURE

- **The test fixture is a fixed (hard-wired) substructure composed of nonmodifiable wires and nonmodifiable electrical components.**
- **Its purpose is to provide a means for testing another electrical substructure, namely the embryo.**
- **A test fixture has one or more ports that enable an embryo to be embedded into it.**
- **The test fixture provides access to the circuit's external input(s) and outputs and permits probing of the circuit's output.**

TWO PARTS OF AN INITIAL CIRCUIT

THE TEST FIXTURE - CONTINUED

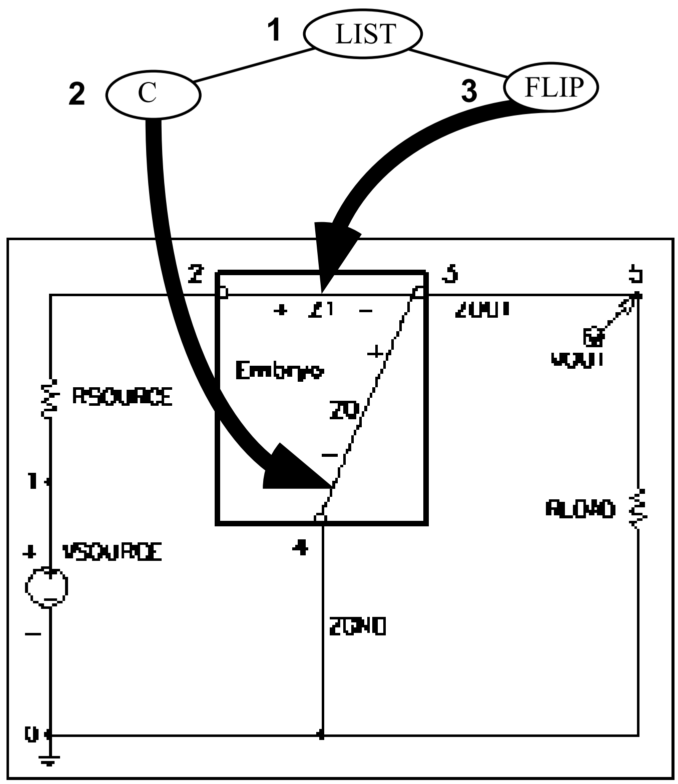
- **A test fixture typically incorporates certain fixed elements that are required to test the type of circuit being designed. For example, the test fixture often contains a source resistor reflecting the reality that all sources have resistance and a load resistor representing the load that must be driven by the output.**
- **The distinctive feature of the test fixture is that it contains no modifiable wires and no modifiable components.**

DEVELOPMENTAL PROCESS

- **In the initial circuit, the test fixture encases only the embryo.**
- **The developmental process operates on the embryo (not the test fixture)**
- **The developmental process applies functions in the circuit-constructing program tree to certain designated elements of the embryo (and its successors).**
- **The functions in the program tree side-effect the embryo (and its successors during the developmental process).**
- **The developmental process ends when the program tree is fully executed.**
- **After the embryo is fully developed, the test fixture encases the nontrivial substructure that is developed from the embryo.**

WRITING HEADS

ONE-INPUT, ONE-OUTPUT INITIAL CIRCUIT WITH TWO WRITING HEADS ASSOCIATED WITH THE TWO MODIFIABLE WIRES (Z0 AND Z1) OF THE EMBRYO



COMPONENT-CREATING FUNCTIONS

TWO-LEADED

- **Resistor `R` function**
- **Capacitor `C` function**
- **Inductor `L` function**
- **Diode `D` function**
- **`TWO_LEAD_OPAMP` function**
- **Digital `NOT` function (inverter)**

THREE-LEADED

- **Transistor `QT` function**
- **`THREE_LEAD_OPAMP` function**
- **Digital `AND`, `OR`, `NAND`, `NOR` functions**

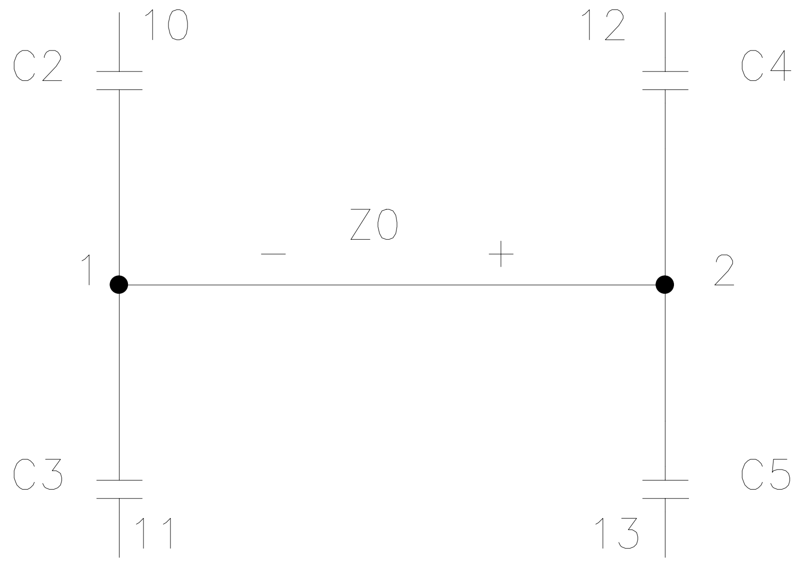
FOUR-LEADED

- **Transformer `TRANFORMER` function**

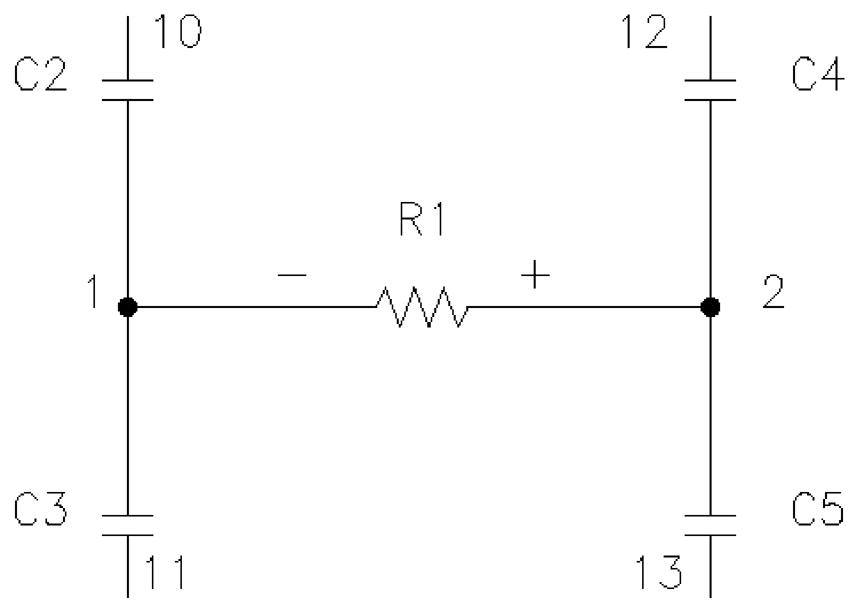
FIVE-LEADED

- **`FIVE_LEAD_OPAMP` function**

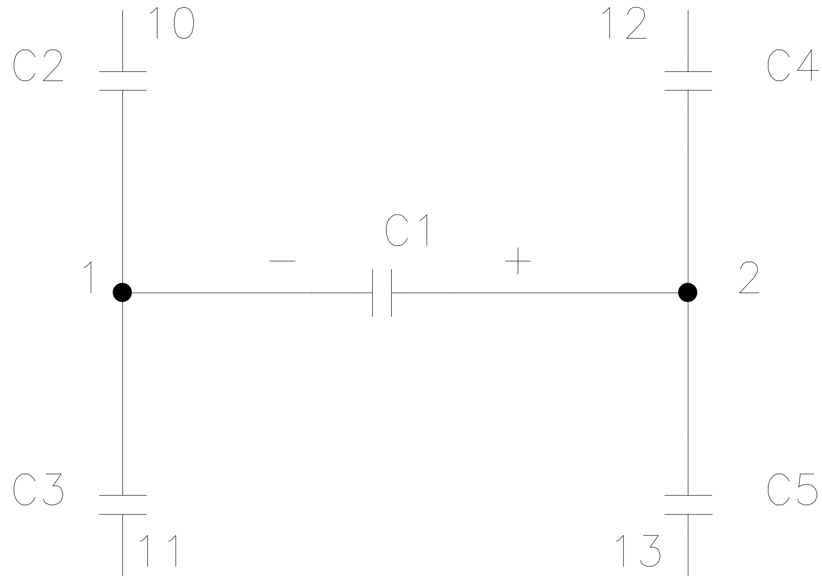
A PORTION OF A CIRCUIT CONTAINING A MODIFIABLE WIRE Z0



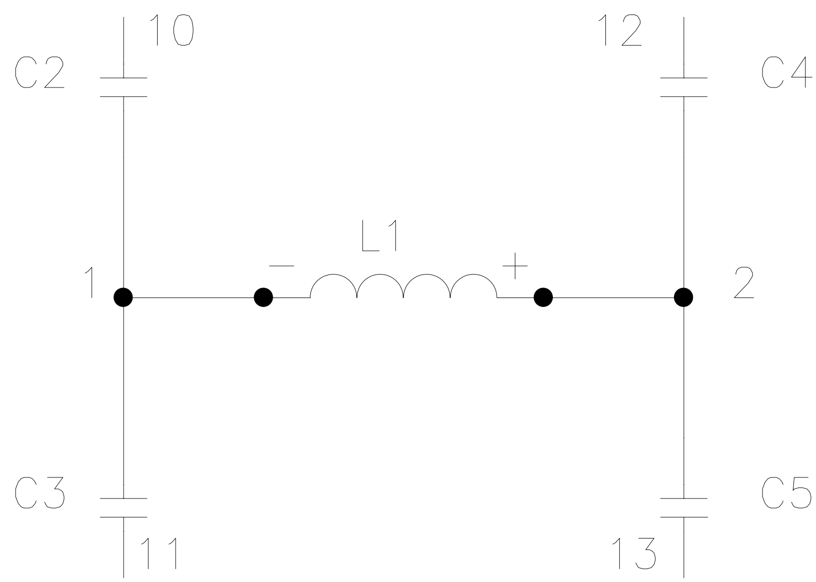
RESULT AFTER R FUNCTION



RESULT AFTER C FUNCTION



RESULT AFTER L FUNCTION



NETLIST BEFORE EXECUTION OF R FUNCTION

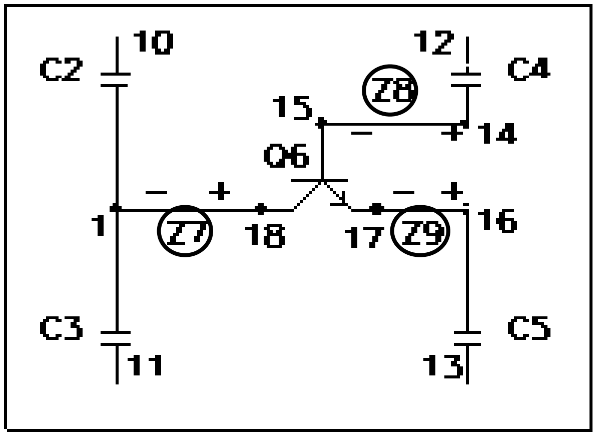
```
C2 1 10
C3 1 11
Z0 2 1
C4 2 12
C5 2 13
```

Note: By convention, the first-listed node is the node connected to the positive lead of a two-leaded component

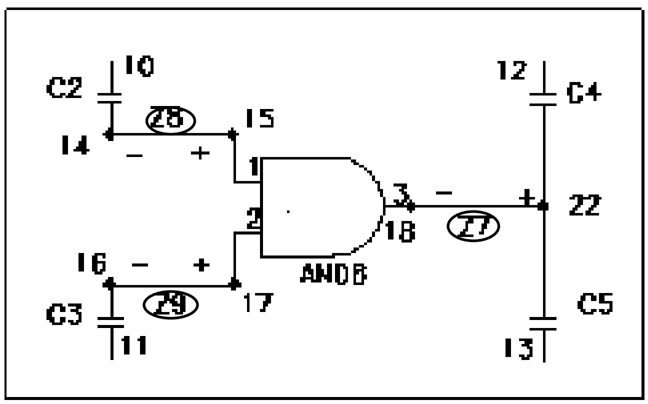
NETLIST AFTER EXECUTION OF R FUNCTION CREATING A 5-Ω RESISTOR

```
C2 1 10
C3 1 11
R1 2 1 5ohms
C4 2 12
C5 2 13
```

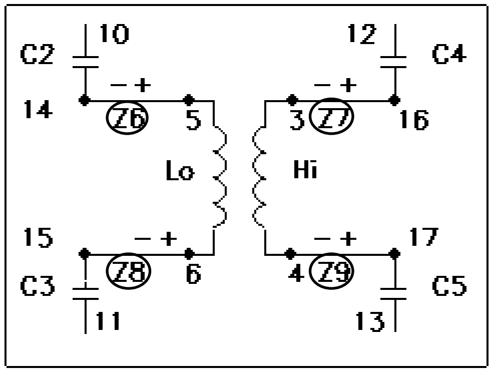
RESULT AFTER QT0 FUNCTION



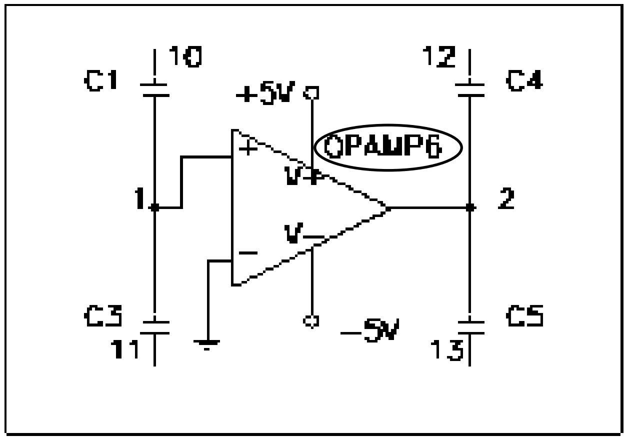
RESULT AFTER AND0 DIGITAL GATE



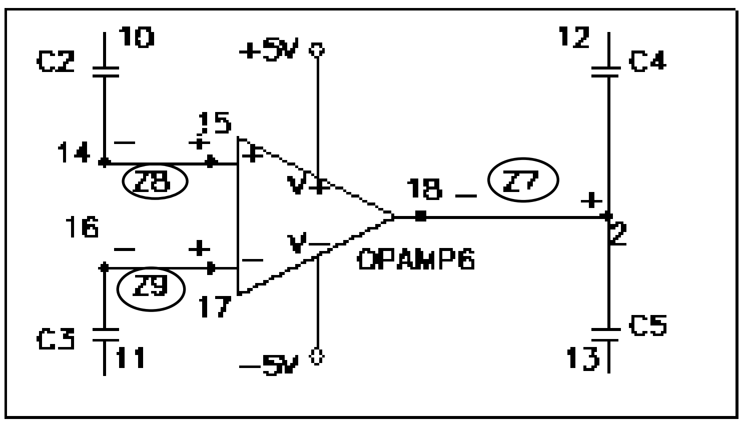
RESULT AFTER `TRANFORMER0` FUNCTION



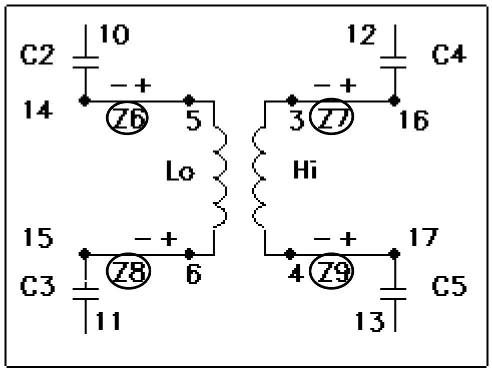
RESULT AFTER TWO_LEAD_OPAMP1



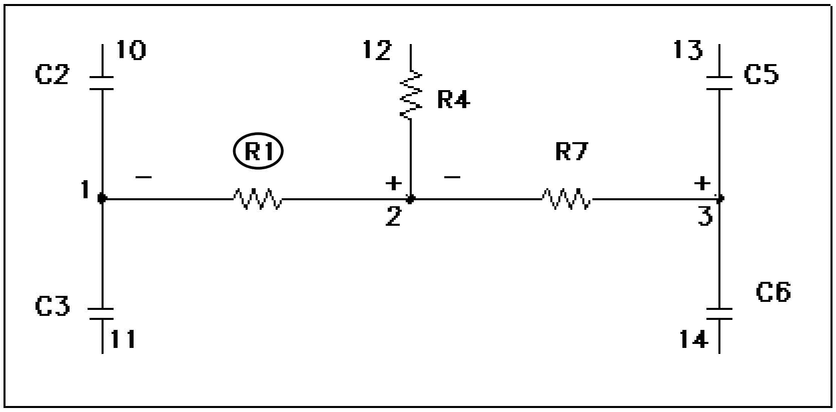
RESULT AFTER THREE_LEAD_OPAMP1 FUNCTION



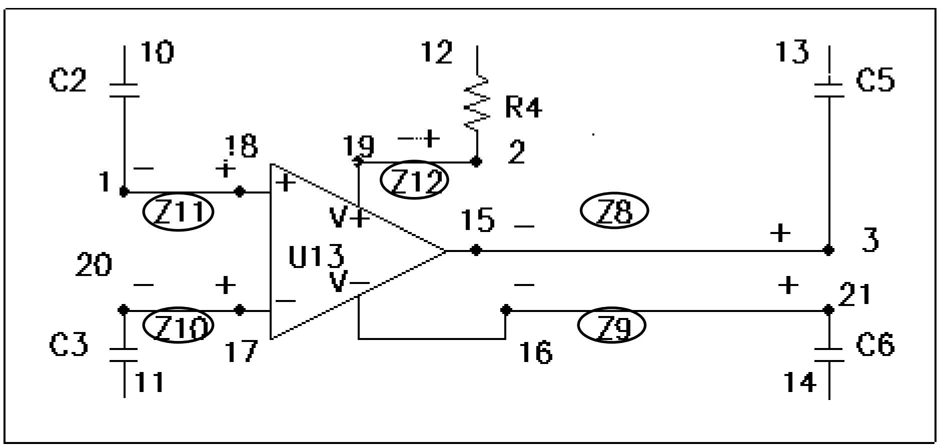
RESULT AFTER TRANFORMER0 FUNCTION



A PORTION OF A CIRCUIT CONTAINING A RESISTOR R1



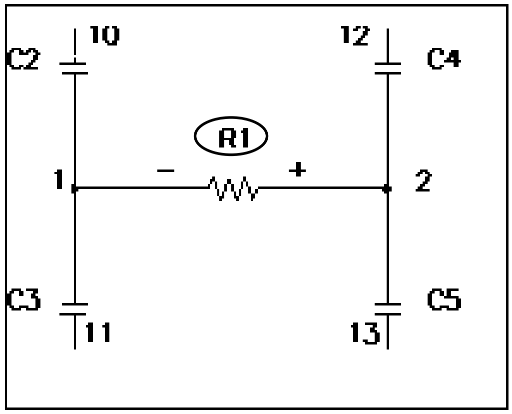
RESULT AFTER FIVE_LEAD_OPAMP3



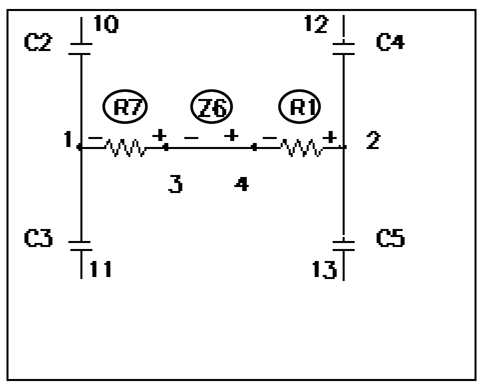
TOPOLOGY-MODIFYING FUNCTIONS

- **SERIES division function**
- **PARALLEL0 and PARALLEL1 parallel division functions**
- **STAR division function**
- **TRIANGLE division function**
- **VIA function**
- **FLIP function**

A CIRCUIT CONTAINING A RESISTOR R1



AFTER THE SERIES FUNCTION



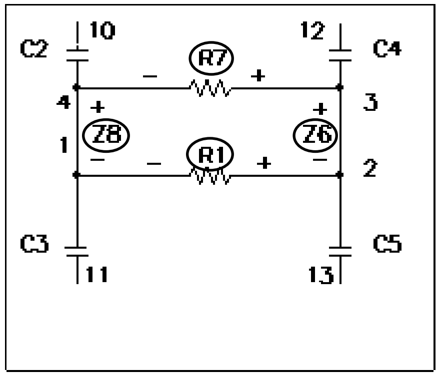
NETLIST WITH R1

```
C2 1 10
C3 1 11
R1 2 1 5ohms
C4 2 12
C5 2 13
```

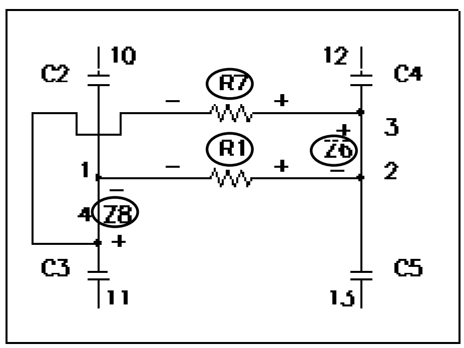
NETLIST AFTER SERIES FUNCTION

```
C2 1 10
C3 1 11
R1 2 4 5ohms
Z6 4 3
R7 3 1 5ohms
C4 2 12
C5 2 13
```

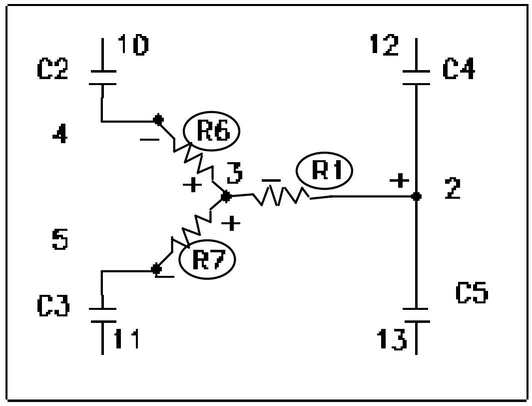
AFTER `PARALLEL0` PARALLEL DIVISION



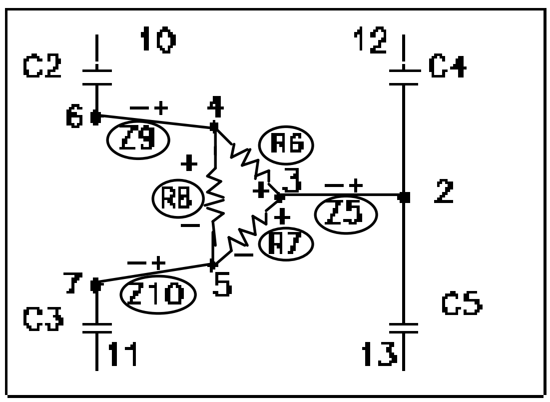
AFTER `PARALLEL1` PARALLEL DIVISION



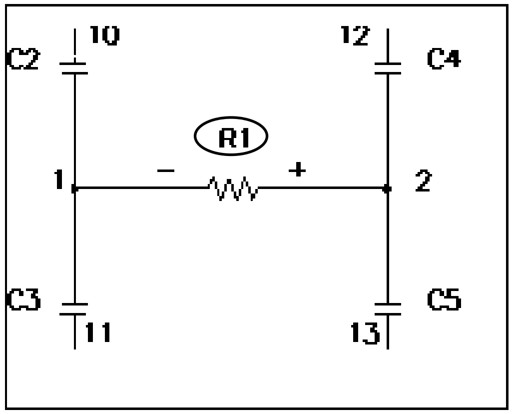
AFTER **STAR1** FUNCTION



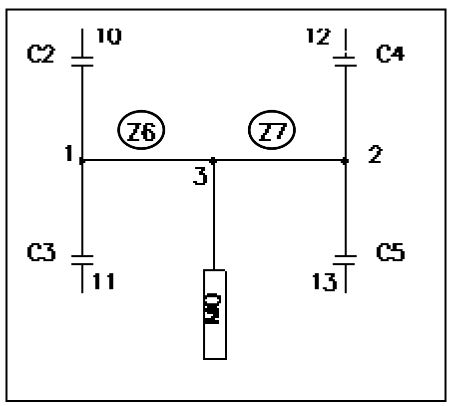
AFTER **TRIANGLE1** FUNCTION



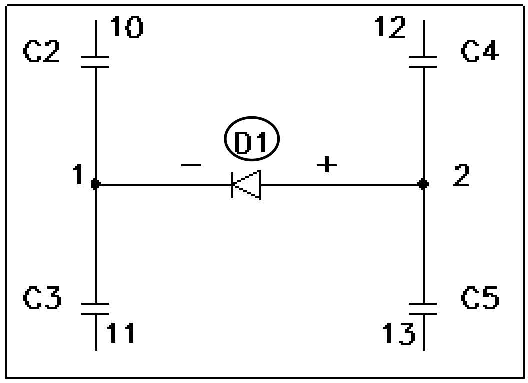
A CIRCUIT CONTAINING A RESISTOR R1



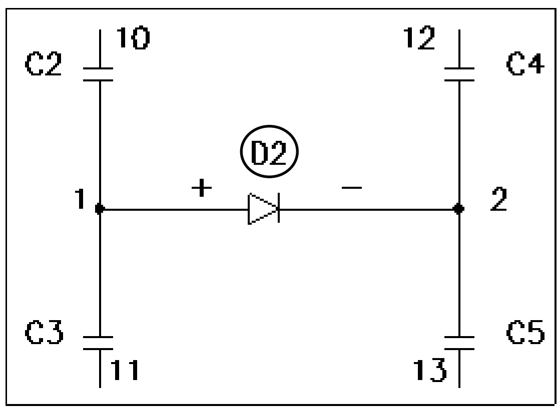
AFTER VIA0 FUNCTION



A CIRCUIT CONTAINING A DIODE D1



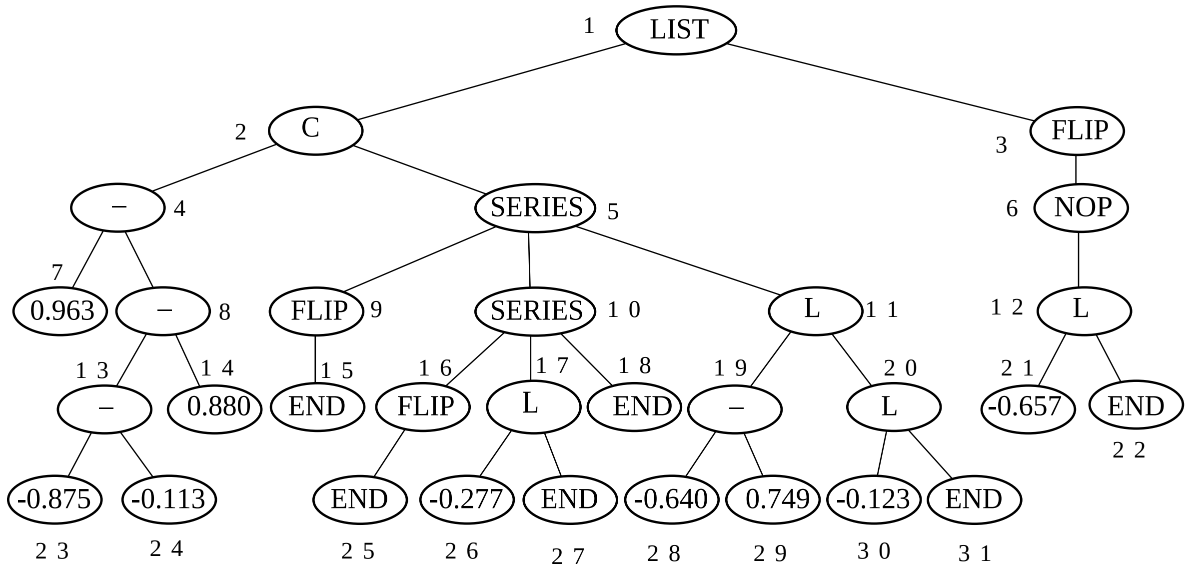
AFTER THE FLIP FUNCTION



DEVELOPMENT-CONTROLLING FUNCTIONS

- **`END` function**
- **`NOP` (No Operation) function**

A CIRCUIT FROM A CIRCUIT-CONSTRUCTING PROGRAM TREE



HIGHLIGHTED ARITHMETIC-PERFORMING SUBTREES

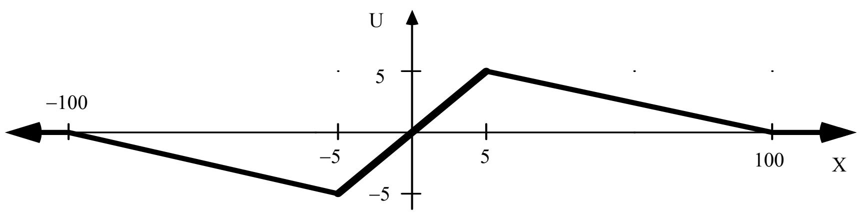
(LIST (C **(– 0.963 (– (– -0.875 -0.113) 0.880)**) (series (flip end) (series (flip end) (L **-0.277** end) end) (L **(– -0.640 0.749)** (L **-0.123** end))))
(flip (nop (L **-0.657** end))))

FUNCTIONS AND TERMINALS FOR ARITHMETIC-PERFORMING SUBTREES

Name	Short Description	Arity
$\Re$	Random constants	0
+	Addition	2
−	Subtraction	2
ADF0, ADF1,...	Automatically defined function 0, etc.	Vario us
ARG0, ARG1,...	Dummy argument 0 (formal parameter 0), etc.	0

LOGARITHMIC INTERPRETATION OF ARITHMETIC-PERFORMING SUBTREES

- **The arithmetic-performing subtree produces floating-point number *X*.**
- ***X* is used to produce an intermediate value *U* in the range of –5 to +5.**



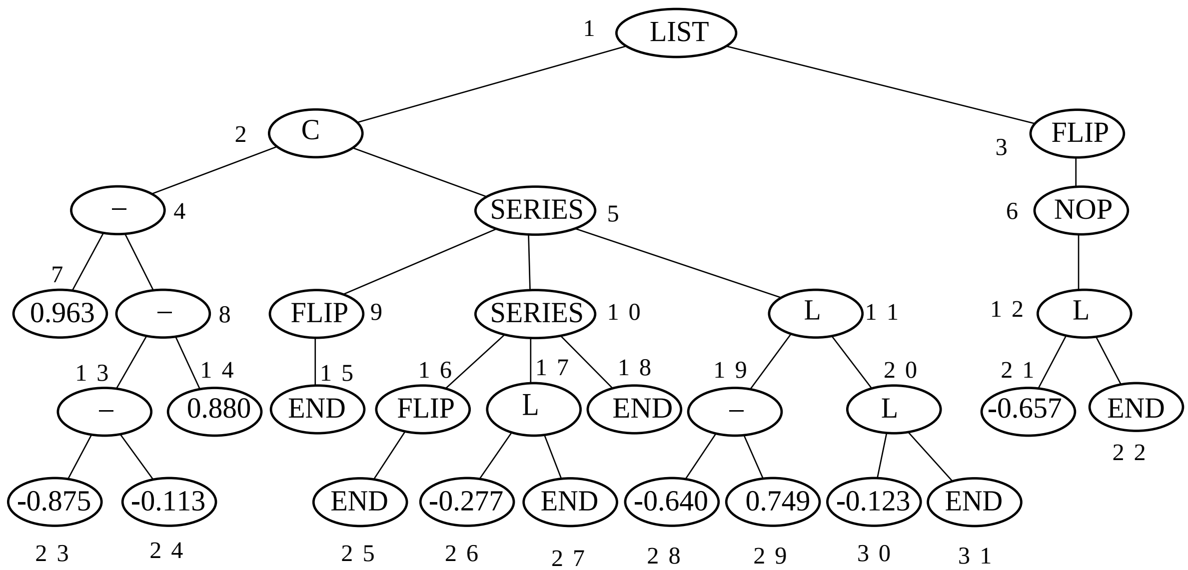
- If the return value *X* is between –5.0 and +5.0, an intermediate value *U* is set to the value *X* returned by the subtree.
- If the return value *X* is less than –100 or greater than +100, *U* is set to a saturating value of zero.
- If the return value *X* is between –100 and –5.0, *U* is found from the straight line connecting the points (–100, 0) and (–5, -5).
- If the return value *X* is between +5.0 and +100, *U* is found from the straight line connecting (5, 5) and (100, 0).

POTENTIAL FUNCTIONS AND TERMINALS FOR AUTOMATICALLY DEFINED FUNCTIONS

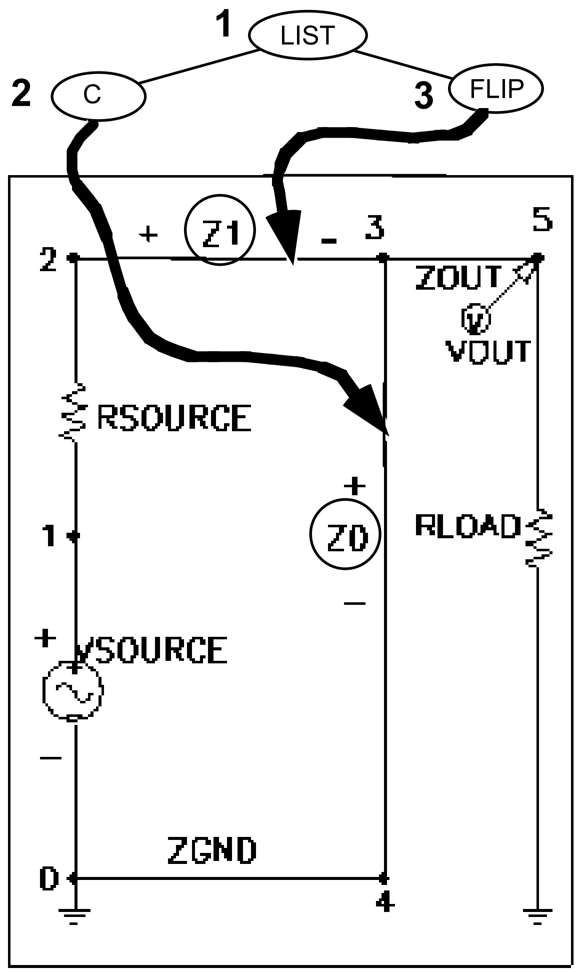
Name	**Short Description**	**Range of Arity**
`ADF-i`	Automatically defined function *i*	0 to MAX_{adf}
`ARG-i`	Dummy variable *i* (formal parameter) of automatically defined function(s)	0 to MAX_{arg}

DEVELOPMENT OF A CIRCUIT FROM A CIRCUIT-CONSTRUCTING PROGRAM TREE AND THE INITIAL CIRCUIT

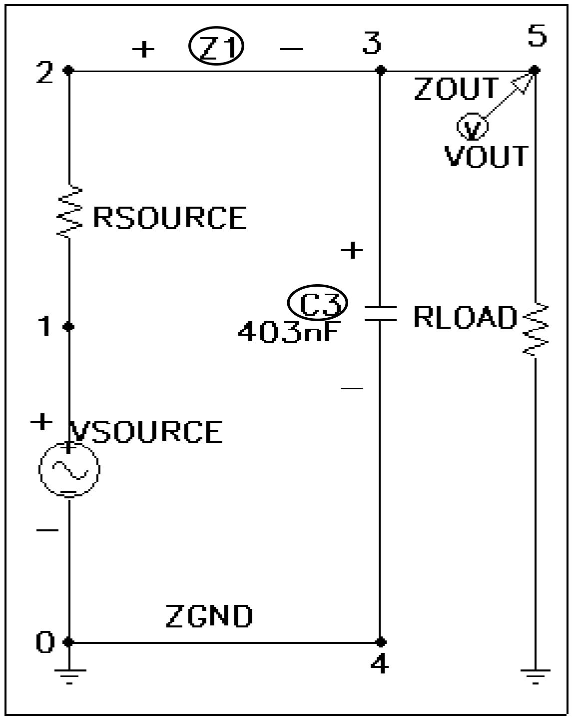
```
(LIST (C (– 0.963 (– (– -0.875 -0.113)
0.880)) (series (flip end) (series (flip
end) (L -0.277 end) end) (L (– -0.640
0.749) (L -0.123 end)))) (flip (nop (L -
0.657 end)))))
```



INITIAL CIRCUIT WITH TWO WRITING HEADS



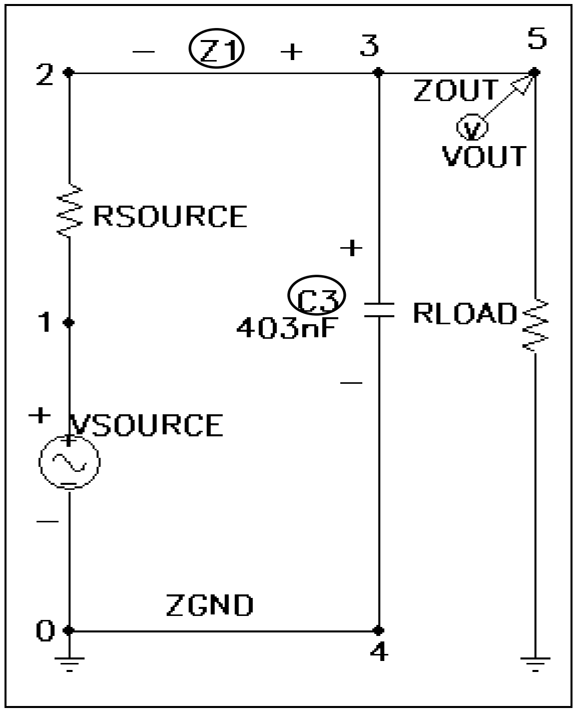
RESULT OF THE C (2) FUNCTION



(LIST **<u>(C (– 0.963 (– (– -0.875 -0.113) 0.880))</u>** (series (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (L -0.123 end)))) (flip (nop (L -0.657 end)))))

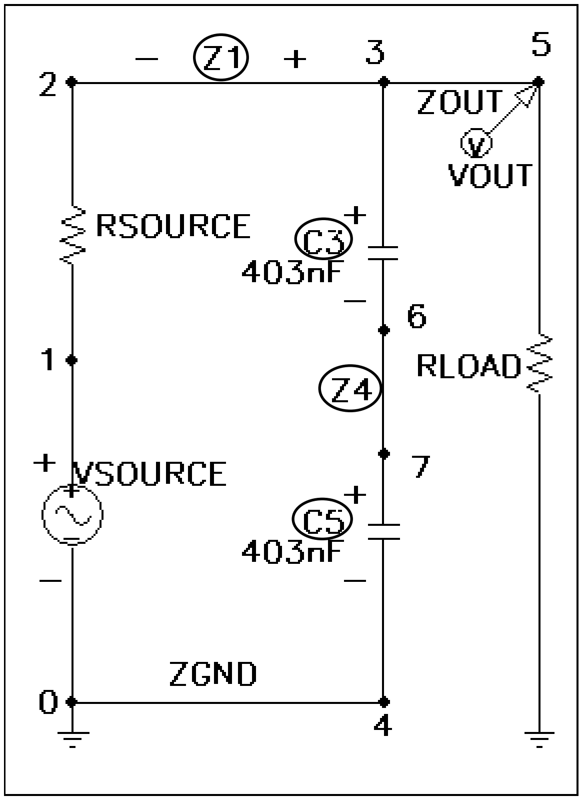
NOTE: Interpretation of arithmetic value

RESULT OF THE FLIP (3) – IN 2nd RESULT-PRODUCING BRANCH



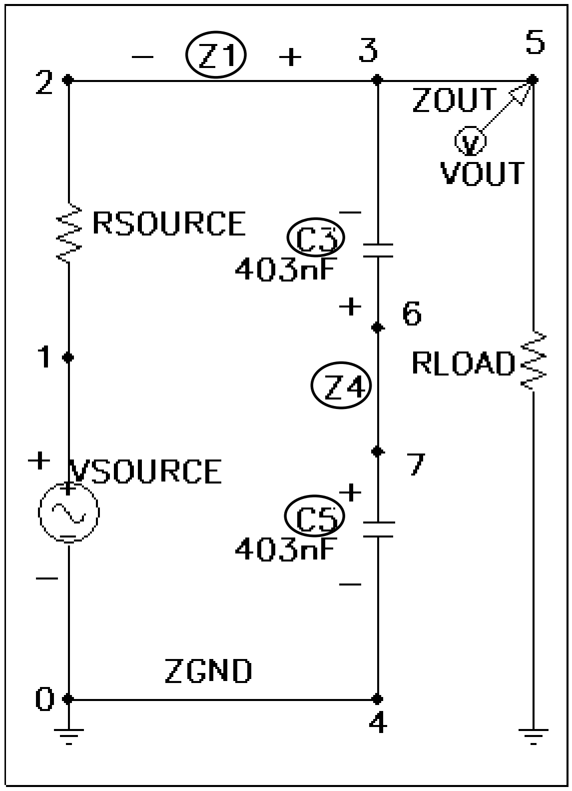
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (series (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (L -0.123 end)))) (**flip** (nop (L -0.657 end)))))

RESULT OF SERIES (5) FUNCTION



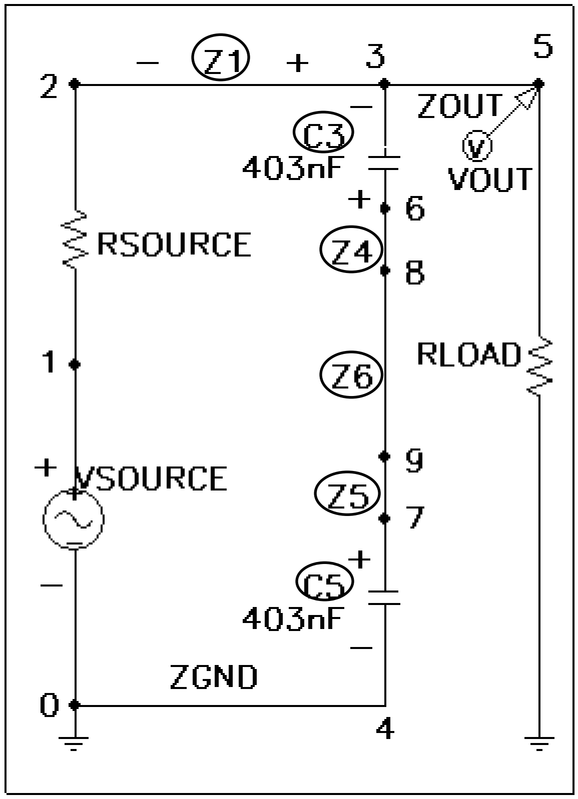
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (**series** (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (L -0.123 end)))) (flip (nop (L -0.657 end)))))

RESULT OF THE **FLIP** (9) FUNCTION



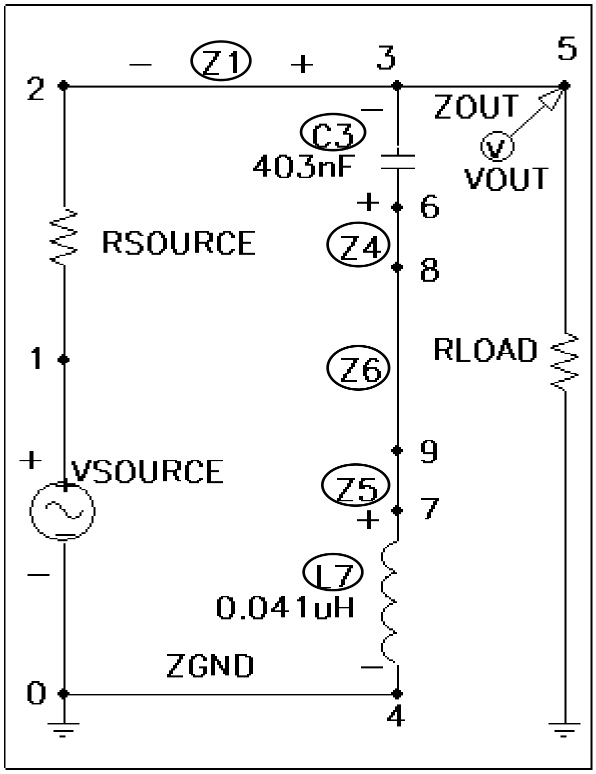

```
(LIST (C (– 0.963 (– (– -0.875 -0.113)
0.880)) (series (flip end) (series (flip
end) (L -0.277 end) end) (L (– -0.640
0.749) (L -0.123 end)))) (flip (nop (L -
0.657 end)))))
```

RESULT OF SERIES (10) FUNCTION



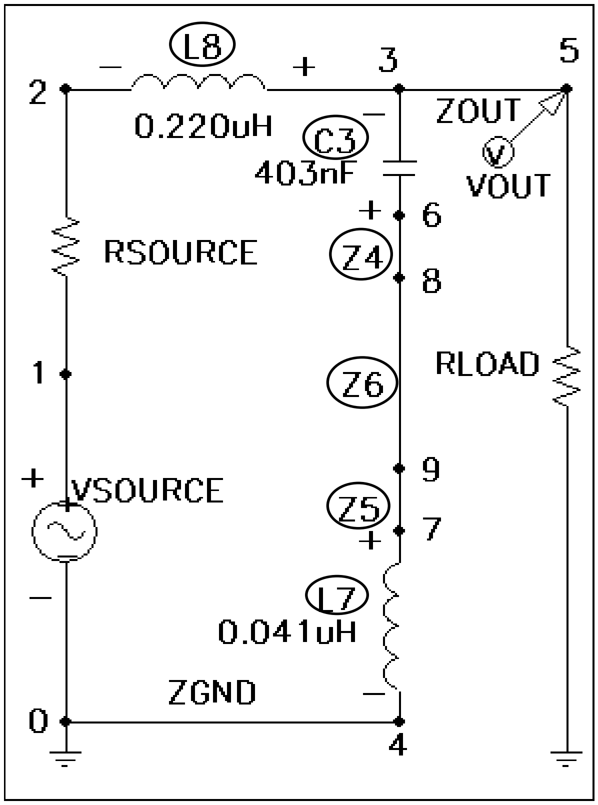

```
(LIST (C (– 0.963 (– (– -0.875 -0.113)
0.880)) (series (flip end) (series (flip
end) (L -0.277 end) end) (L (– -0.640
0.749) (L -0.123 end)))) (flip (nop (L -
0.657 end)))))
```

RESULT OF L (11) FUNCTION



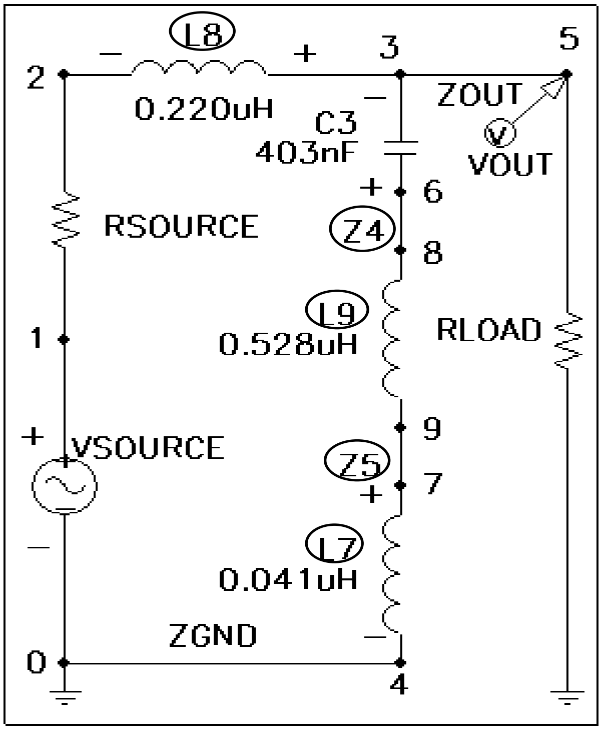
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (series (flip end) (series (flip end) (L -0.277 end) end) **(L (– -0.640 0.749)** (L -0.123 end)))) (flip (nop (L -0.657 end)))))

RESULT OF L (12) FUNCTION



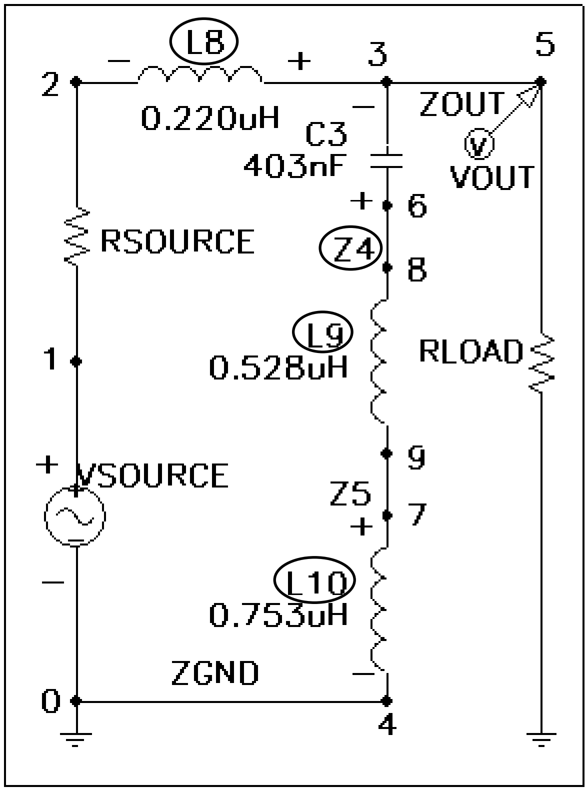
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (series (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (L -0.123 end)))) (flip (nop (**L -0.657** end)))))

RESULT OF L (17) FUNCTION



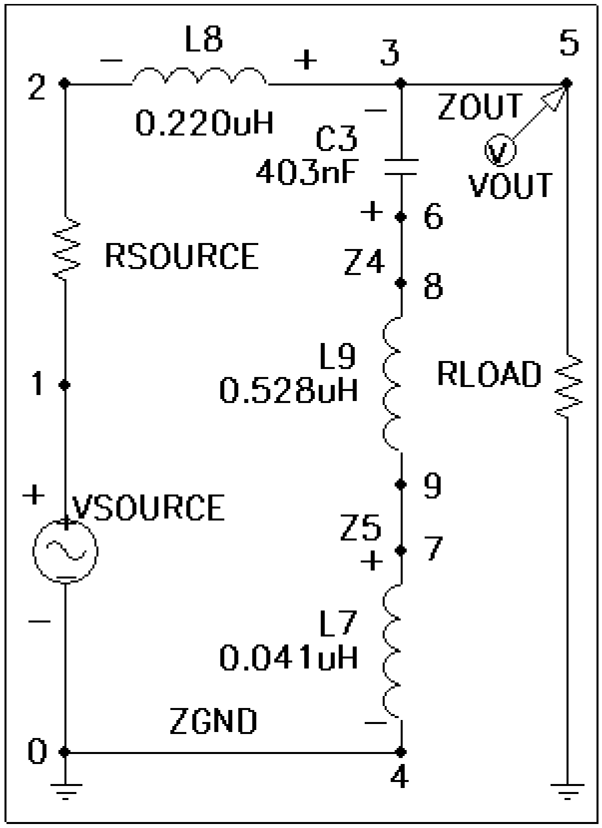
(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (series (flip end) (series (flip end) **(L -0.277** end) end) (L (– -0.640 0.749) (L -0.123 end)))) (flip (nop (L -0.657 end)))))

RESULT OF L (20) (OVERWRITING)

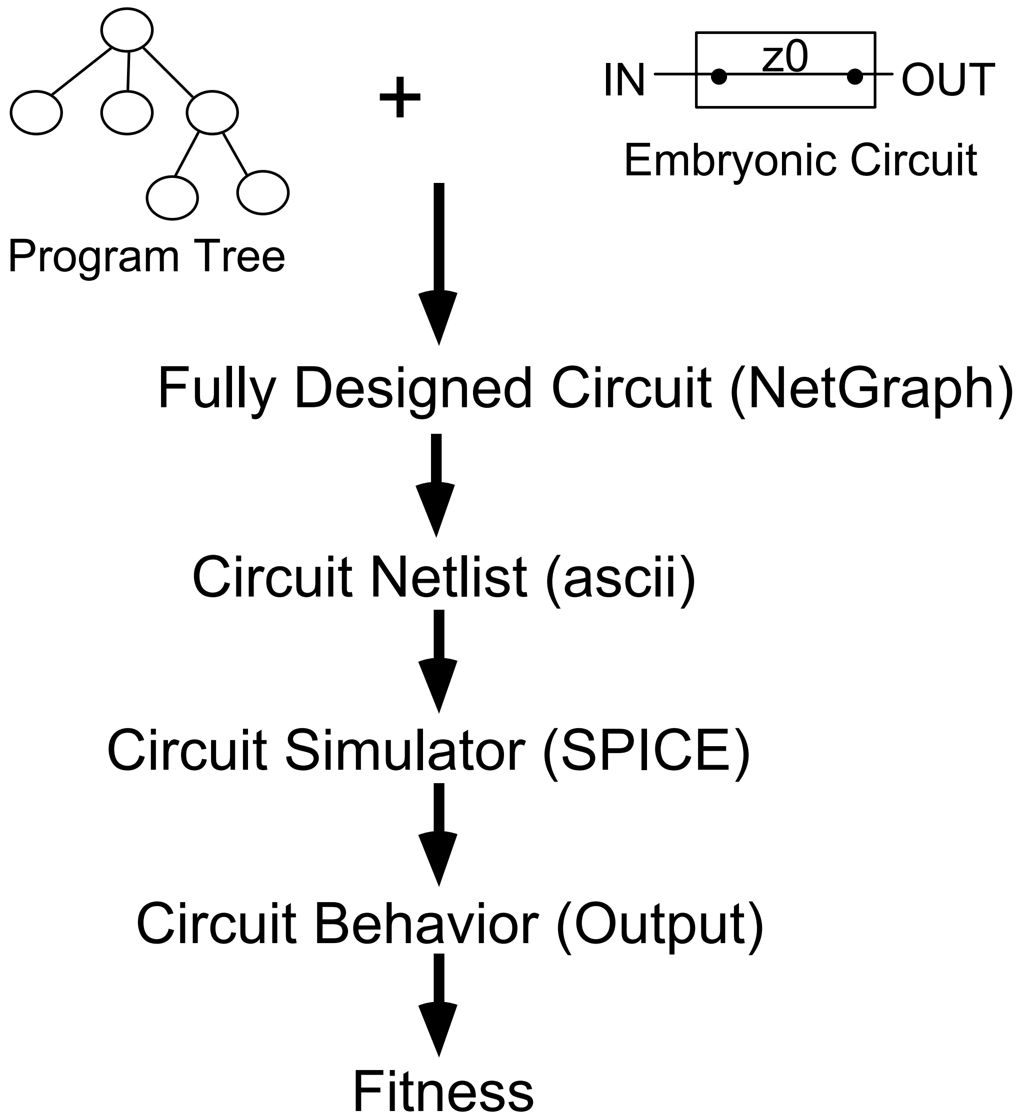


(LIST (C (– 0.963 (– (– -0.875 -0.113) 0.880)) (series (flip end) (series (flip end) (L -0.277 end) end) (L (– -0.640 0.749) (**L -0.123** end)))) (flip (nop (L -0.657 end)))))

THE FULLY DEVELOPED EXAMPLE CIRCUIT (BOG 0 - LPF)



FITNESS MEASUREMENT PROCESS



SPICE CIRCUIT SIMULATOR

- **Developed at Univ. of California - Berkeley**
- **217,000 lines of C code**
- **Hundreds of thousands of users**
- **Available in many forms from various sources (e.g., PSPICE, HSPICE)**
- **Input required by SPICE**
  - Netlist
  - SPICE Commands
  - Models
- **Types of analysis produced by SPICE**
  - Fourier
  - AC Sweep
  - DC Sweep
  - Transient
  - Operating point
  - Sensitivity
  - Distortion
  - Noise
  - Pole-Zero
  - Transfer function

SPICE NETLIST

```
Circuit Must Have a Title
V0         2  0    DC 0V
QD2        3  3  4 Q2P3906
R6         1  0    1.00e+00K
R7         2  1    1.00e+00K
QNC9       5  1  1 Q2P3906
QGE12      1  6  0 Q2N3904
QD13       1  1  4 Q2P3906
QNC15      5  3  7 Q2P3906
Q17        7  6  0 Q2N3904
VNEGQ999   5  0    DC -15V
.MODEL    Q2P3906PNP    Bf=180.7
Br=4.977
.MODEL    Q2N3904NPN    Bf=416.4
Br=0.7371
.DC V0 -0.25 0.251 0.025
.PLOT DC V(1)
.OPTIONS NOPAGE NOMOD
.END
```