CIRCUIT THEORY AND APPLICATIONS, VOL. 8, 127–144 (1980)

PROPAGATION OF CONSTRAINTS APPLIED TO CIRCUIT SYNTHESIS*

JOHAN DE KLEER† AND GERALD JAY SUSSMAN
Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, Mass. U.S.A.

SUMMARY

A major component in the process of design is synthesis, the determination of the parameters of the parts of a network given desiderata for the behaviour of the network as a whole. Traditional automated synthesis techniques are either restricted to small, precisely defined classes of circuit functions for which exact mathematical methods exist or they depend upon numerical optimization methods in which it is difficult to determine the basis for any of the answers generated and their relations to the design desiderata and constraints. We are developing a symbolic computer-aided design tool, SYN, which can be of assistance to an engineer in the synthesis of a large class of circuits. The symbolic methods produce solutions which are clear and insightful. The dependence of each parameter on the individual design desiderata and circuit constraints can be easily traced.

INTRODUCTION

A major component in the process of design is synthesis, the determination of the parameters of the parts of a network given desiderata for the behaviour of the network as a whole. Traditional automated synthesis techniques are either restricted to small, precisely defined classes of circuit functions for which exact mathematical methods exist or they depend upon numerical optimization methods in which it is difficult to determine the basis for any of the answers generated and their relations to the design desiderata and constraints[1].‡ We have developed a symbolic computer-aided design tool, SYN, which can assist an engineer in the synthesis of a large class of circuits. The symbolic methods produce solutions which are clear and insightful. The dependence of each parameter on the individual design desiderata and circuit constraints can be easily traced.

It is an obvious idea to try to determine the component values by solving the set of equations and inequalities which result from matching a symbolic analysis of the circuit with the given design desiderata. Unfortunately, this is algebraically infeasible in general. A complete symbolic analysis of even simple circuits containing non-linear components is usually difficult[2]. But even for circuits whose behaviour is linear in voltages and currents, the equations are non-linear in the component parameter values.

Our synthesis aid is based on analysis by propagation of constraints[3]. This analysis method guides the use of symbolic algebraic methods[4] in combining constraints which describe circuit elements and their interconnections to determine the behaviour of a circuit. In this paper we show how propagation analysis can be inverted to determine constraints on the individual parts from the desired behaviour of the circuit. The method is based on the observation that locally, analysis and synthesis are very similar: The problem of finding the resistance which permits a given current flow at a given potential is equivalent to the problem of determining the current that flows given a resistance and a potential.

Our method is successful for several reasons. It does not try to invert a complete analysis. The method of propagation of constraints deals with only a small part of the problem at a time. It is an incremental deductive method which first solves whatever subproblems can be solved easily. After picking off the easiest

* This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's research is provided in part by the National Science Foundation under Grant MCS77–04828.
† Present address: Xerox, Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A.
‡ Numbered superscripts relate to 'Notes and relation to other work' on p. 143. Citations of References are enclosed in brackets, e.g. [TABLEAU] etc.

0098–9886/80/0208–0127$01.00

Received 25 January 1979
Revised 5 October 1979

parts, the remainder has been reduced, leaving new easy parts to be picked off next. If there remain unsolved parts of the problem because there is not enough information to uniquely specify a solution or because the remaining constraints are too hard to solve, the method can accept advice from the user which will break the impasse. This advice can be formalized as additional algebraic constraints which encode 'rules of thumb' similar to those used by human designers. In addition, our method can use information deduced from several different models of a circuit, each constraining only a portion of its behaviour.

AN EXAMPLE OF SYMBOLIC ANALYSIS

In order to illustrate the method we present a detailed annotated interaction with SYN. After this we will give a general discussion of the methods involved in the techniques illustrated. In the following example, input to the computer is in lower-case, and its response is in upper-case.

In this example we will deal with the cascode amplifier of Figure 1.

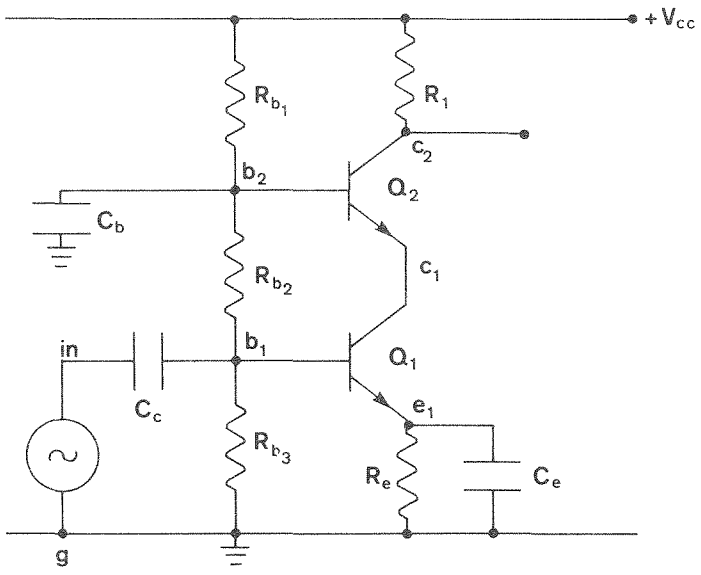

Figure 1. A cascode amplifier

We communicate this circuit diagram to the computer by declaring the nodes in the diagram and specifying the components connected between them.

```
(diagram cascode
        (in vcc g e1 b1 c1 b2 c2)          ; These are the nodes.
        ((bat (battery vcc g))             ; A battery between vcc and g.
         (s-gen (signal-generator in g))
         (cc (capacitor in b1) coupling)
         (cb (capacitor b2 g) bypass)
         (ce (capacitor e1 g) bypass)
         (q1 (transistor e1 b1 c1))        ; q1 is the input transistor.
         (q2 (transistor c1 b2 c2))
         rb1 (resistor vcc b2))
         (rb2 (resistor b2 b1))
```

(rb3 (resistor b1 g))
(rl (resistor vcc c2))
(re (resistor e1 g))))
CASCODE

The capacitors C_c, C_b and C_e are declared to be coupling or bypass capacitors. This teleological information will be used to determine how these components are modelled. Of course, there is other information encoded in the conventional layout of the circuit diagram which SYN has not been told, such as the fact that R_{b1}, R_{b2}, and R_{b3} form a bias divider. This sort of information is extensively used by human engineers in their thinking about circuits, but we are only beginning to understand how it is helpful.

After defining the cascode circuit we can request that SYN consider a new instance of a cascode. This is necessary since a designer may be working with many cascodes at once. The result of this operation is a particular cascode called CASCODE-1.

(cascode)
CASCODE-1

Rather than dealing with the complexities of the circuit as a whole, an engineer simplifies his problem by constructing models which describe aspects of the behaviour of the circuit being designed. A circuit model is a new circuit related to the circuit being modelled in that the voltages and currents in the model represent components of the voltages and currents in the original circuit.

For example, a bipolar junction transistor is a three terminal device with badly non-linear behaviour. The problem can be simplified by decomposing the behaviour of a transistor into several parts. If a transistor is to be used as an amplifier it will be operated in the forward active region. In that region, we can approximate the quiescent behaviour of the transistor by a simple circuit model (see Figure 2). Once we know the

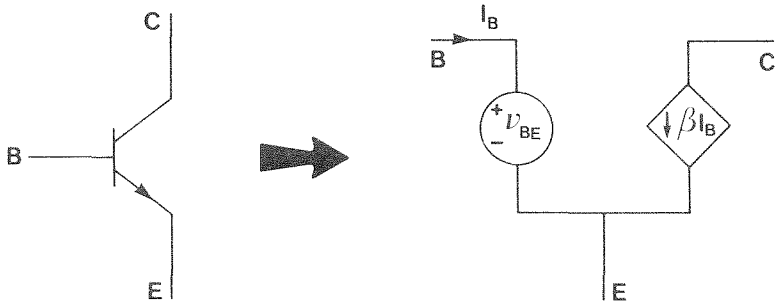

Figure 2. Transistor bias model

quiescent behaviour of a transistor in a circuit, we proceed to consider small deviations from that quiescent behaviour due to the signals being processed. The voltages and currents in the hybrid-pi model of a transistor (see Figure 3) approximate the incremental voltages and currents in the actual transistor.

Parameters in one model may depend upon the behaviour of the circuit described by another model. For example, the small signal transconductance of a transistor depends upon the bias current.[5] In design a desired value of circuit gain may constrain the transconductance. This constraint must eventually be used to constrain the bias current. We will later discuss how one can specify models for devices and the constraints among their parameters.

The models for devices in a circuit can be combined to form a model for the behaviour of the entire circuit. Thus the small signal behaviour of a transistor amplifier may be modelled by a circuit in which all

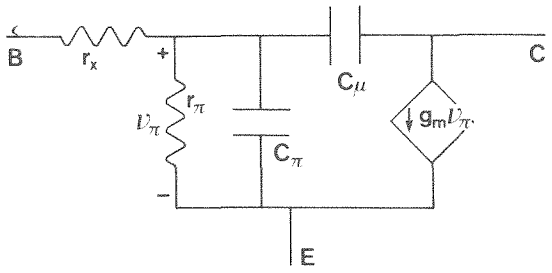

Figure 3. Hybrid-pi incremental transistor model

bias sources are set to zero, and in which the transistors are represented by their hybrid-pi descriptions. The voltages and currents in the model circuit are then an approximation to the deviations of the voltages and currents from their bias values in the modelled circuit.

We now ask SYN to construct various models of the cascode amplifier. First we request an ac model.

(make-model (ac cascode-1))
CASCODE-2

CASCODE-2 is the resulting model circuit. The schematic diagram is shown in Figure 4.

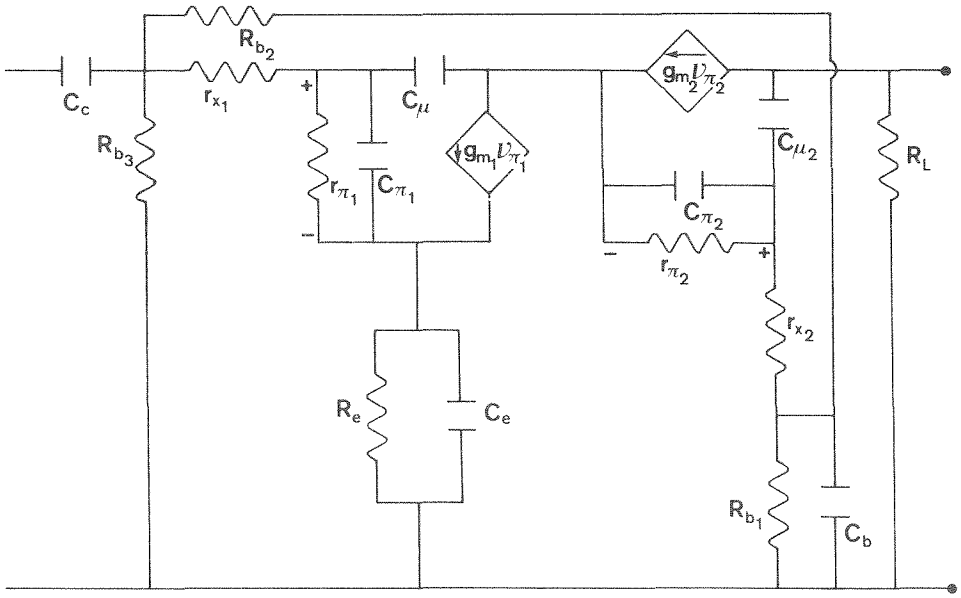

Figure 4. The incremental (ac) model of CASCODE-1

Often it is useful to construct a model of a model which approximates its behaviour in a limited context. For example, the midband behaviour of an amplifier may be approximated by removing all coupling, bypass, and parasitic capacitances from a small signal model. The amplifier we are designing will be specified to have a particular midband gain, so we request a midband of the incremental model.

(make-model (midband ac cascode-1))
CASCODE-3

CASCODE-3 is the midband model that was constructed (see Figure 5). The parasitic capacitances (C_π and C_μ of each transistor) have been modelled by open circuits and the bypass and coupling capacitances have been modelled as short circuits. We next proceed to define for SYN what we mean by the small signal gain of the amplifier in the midband model.

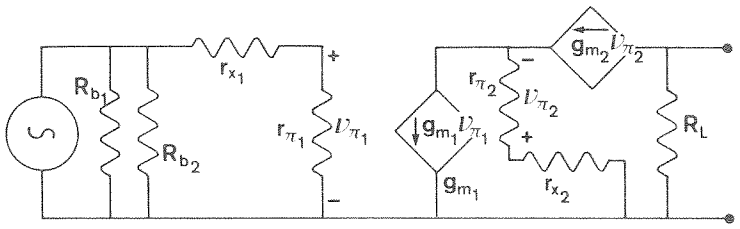

Figure 5. The midband model of CASCODE-1

```
(constraint cascode-gain (midband ac cascode-1) (gain)
          ((vi (voltage in g))
           (vo (voltage c2 g))
           (g (parameter-value gain)))
          (+g (: vo vi)))
CASCODE-GAIN
```

A constraint definition has a name, a model to which it applies, a list of new variables used in the constraint expression, a definition of the variables being related by the expression, and finally the algebraic expression which is constrained to have value zero. Thus, in this case, the midband gain of the cascode amplifier is defined to be the negation of the ratio of the output voltage to the input voltage. (This is convenient because the cascode is an inverting amplifier.)

Now that the gain is defined, we can ask SYN for the value of the gain. Because the size of the symbolic expressions increases very quickly with the number of symbolic quantities they contain, we cannot expect SYN to be able to perform a complete symbolic analysis for a case much more complex than this one. (We will discuss the algebraic limitations later.) However, here we are in luck.

```
(what-is (parameter-value gain cascode-1))
```

$$\frac{\text{R-PI\_Q1 GM\_Q1 R-PI\_Q2 GM\_Q2 RL}}{\text{(R-X\_Q1 + R-PI\_Q1)R-PI\_Q2 GM\_Q2 + R-X\_Q1 + R-PI\_Q1}}$$

Some of the specifications of the circuit will be in terms of the bias (dc) model, so we request the construction of that model as well.

```
(make-model (dc cascode-1))
CASCODE-4
```

The resistances of the various resistors appearing in the original circuit appear in the individual models as well. This is another way in which information is shared among the models. Constraints imposed on the aspects of the behaviour of the circuit represented by each of these models will combine to constrain the values of the shared parameters. In fact, constructing the bias model caused SYN to refine its view of the gain of the cascode amplifier.

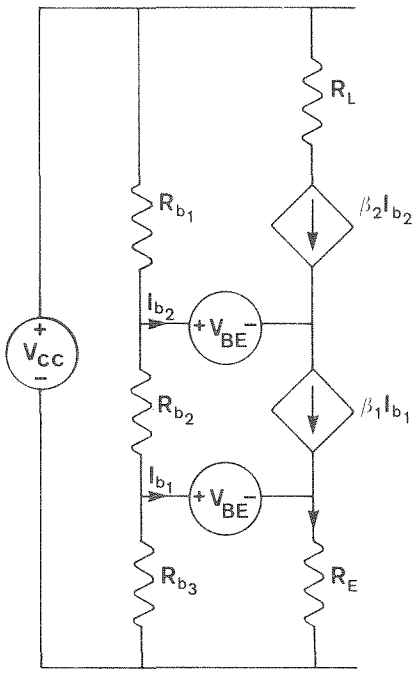

Figure 6. The bias (dc) model of CASCODE-1

(what-is (parameter-value gain cascode-1))

$$\frac{\text{R-PI\_Q1 GM\_Q2 RL}}{\text{R-X\_Q1} + \text{R-PI\_Q1}}$$

The new gain expression is considerably simpler than the previous one, in fact, neither gm_1 nor r-pi_2 appears in this new expression. Looking only at the schematic diagram of the midband model (see Figure 5), one would expect the gain expression to include these variables. In order to help understand what happened we may examine gm_1.

(what-is (parameter-value gm q1 cascode-1))

$$\frac{\text{R-PI\_Q2 GM\_Q2} + 1}{\text{R-PI\_Q2}}$$

The value of gm_1 was determined in terms of other parameters by consideration of the dc model. The critical facts that SYN knows which it used to make this deduction are:

1. The bias current into the collector of q1 is the current coming out of the emitter of q2.
2. The gm (transconductance) of a transistor is proportional to its collector bias current.
3. The product of the gm and r-pi of a transistor is beta, the ratio of the collector bias current to the base bias current.
4. The sum of the bias currents into the base, emitter, and collector of a transistor are always zero.

If this value for gm_1 is substituted into the original gain expression, r-pi_2 cancels out. A more extensive analysis of the dc bias model would have determined even more about the gain, but SYN is currently concentrating on the signal characteristics of the system. Interactions of this kind demonstrate the extensive communications between models that are necessary for circuit synthesis.

RULES OF THUMB FOR SYNTHESIS

One value of our synthesis system lies in its ability to combine constraints inherent in the structure of the circuit with constraints imposed by the user on the behaviours of the models of the circuit to produce constraints on the device parameters. In the next few expressions we enter several rules of thumb for stable biasing of this circuit.

The first hint is that the current in the bottom resistor of the base bias voltage divider chain be one tenth of the current in the emitter bias resistor. The rationale behind this rule (which SYN doesn't know) is that we want the base bias chain current small so that the input impedance of the circuit is high, but we want it large enough so that variations in base currents into the transistors due to variations in transistor beta do not cause significant variations in the base voltages set up by the bias voltage divider.

```
(constraint bias-rule-1 (dc cascode-1) ()
          ((ibb (current # 1 rb3))
           (ie (current # 1 re)))
          (- ie (* ibb 10)))
BIAS-RULE-1
```

Next, we constrain the voltage across the emitter bias resistor to be 5 times the specified intrinsic voltage drop of the transistor's base–emitter junction. This rule is to ensure the bias stability of the circuit by making the voltage at the top of the emitter resistor (which determines the emitter current of the transistor) big enough to be relatively independent of variations in intrinsic voltage drop of the transistor's base–emitter junction.

```
(constraint emitter-bias (dc cascode-1) ()
          ((ve (voltage e1 g))
           (vbe (parameter-value vbe q1)))
          (- (* 5 vbe) ve))
EMITTER-BIAS
```

Finally, we specify that the base of the output transistor of the cascode be suspended halfway between the positive supply rail and the bias voltage for the base of the input transistor.

```
(constraint separation (dc cascode-1) ()
          ((v1 (voltage b2 b1))
           (v2 (voltage vcc b2)))
          (- v1 v2))
SEPARATION
```

At this point SYN has computed some of the consequences of these constraints and will be able to use them later. In a purely analytic context the values of circuit parameters are unconstrained. Any resistor may have any resistance. But here, the constraints inherent in the circuit diagram have been combined with the new constraints bias-rule and emitter-bias to make some circuit parameters depend upon others. Thus, for example, the resistance of rb3 is now linked to the resistance of re:

```
(what-is (parameter-value resistance rb3 cascode-1))
12 RE
```

Other new relationships are not quite so simple. For example, the relationship between various elements of the bias-chain depends upon properties of the transistors which are currently unspecified:

(what-is (parameter-value resistance rb1 cascode-1))

$$\frac{(\text{R-PI\_Q1 R-PI\_Q2 GM\_Q2+11 R-PI\_Q2+R-PI\_Q1 }) \text{ RB2}}{\text{R-PI\_Q1 R-PI\_Q2 GM\_Q2+11 R-PI\_Q2+11 R-PI\_Q1}}$$

These new constraints will be useful in determining the parameter values when more information becomes available.

A SPECIFIC DESIGN

Now that all the general rules of thumb have been specified, we engage in the design of the particular circuit. First we enter some of the detailed specifications of the two transistors we are using. The transistors are identical; with beta = 100, vbe = 0·6 V, and r-x = 50 Ω.

```
(use-parameters (cascode-1)
                ((100 (beta q1)) (0·6 (vbe q1)) (50 (r-x q1))
                 (100 (beta q2)) (0·6 (vbe q2)) (50 (r-x q2))))
USE-PARAMETERS
```

Whenever SYN discovers a constraint among a set of unknowns, it chooses one and solves for it in terms of the others. Any subsequent reference to that variable is then treated as a reference to the expression which was determined to be its value. This informal elimination procedure is equivalent to Gaussian elimination when applied to linear equations, though analysis by propagation of constraints takes advantage of the sparseness of the equations. It produces fewer equations in fewer unknowns than traditional analysis methods. The informal elimination method is not limited to the solution of linear equations. It slowly reduces the set of unknowns in terms of which other values are represented. Indeed, if we now look at SYN's opinion about rb1 we find that the parts of the expression which depended upon the properties of the transistors have simplified to a numerical value.

(what-is (parameter-value resistance rb1 cascode-1))

$$\frac{11211 \text{ RB2}}{12211}$$

We see that rb1 is represented in terms of rb2 only. If we were to ask for the value of rb2 SYN would respond with an uninformative RB2. This is not indicative of a lack of constraints among rb2 and other unknowns; it is just that rb2 is a member of that subset of the set of unknowns in terms of which the other unknowns are described. There are *no* solvable constraints yet available among members of this residual subset.

The gain expression is also somewhat simplified in the light of the new information.

(what-is (parameter-value gain cascode-1))

$$\frac{1200000 \text{ RL}}{10201 \text{ RE}+606000}$$

Next we enter the constraint that the power-supply has 15 V.

```
(constraint vcc-supply (dc cascode-1) ()
           ((vcc (voltage vcc g)))
           (- 15 vcc))
VCC-SUPPLY
```

This constraint on vcc allows SYN to determine rb2 in terms of re.

(what-is (parameter-value resistance rb2 cascode-1))

$$\frac{1919\ RE}{111}$$

If we ask for the resistance of rb1 again, SYN will substitute in this value for rb2.

(what-is (parameter-value resistance rb1 cascode-1))

$$\frac{193819\ RE}{12211}$$

Now we enter the constraint that we desire that the amplifier have a gain of 100.

```
(constraint desired-gain (cascode-1) ()
          ((g (parameter-value gain)))
          (- g 100))
DESIRED-GAIN
```

We ask whether the system can tell us the value of the collector resistor.

(what-is (parameter-value resistance rl cascode-1))

$$\frac{10201\ RE+606000}{12000}$$

All of the resistances are now known in terms of re. In order to determine re, one more constraint is required. We know that the particular transistors we are using operate best at 1 mA.

```
(constraint ic-current (dc cascode-1) ()
          ((i (current c q2)))
          (-  ·001 i))
IC-CURRENT
```

At this point, we see that the emitter resistor is known.

```
(what-is (parameter-value resistance re cascode-1))
2940·88815
```

Thus all the other resistances are known.

```
(what-is (parameter-value resistance rl cascode-1))
2550·5
(what-is (parameter-value resistance rb3 cascode-1))
35290·6577
(what-is (parameter-value resistance rb2 cascode-1))
50842·9224
(what-is (parameter-value resistance rb1 cascode-1))
46679·2236
```

HOW IT WORKS

SYN is based on propagation of constraints. Abstractly, there are *cells* each of which represent an electrically interesting quantity, such as a voltage, current or resistance. Each cell participates in one or more *constraint expressions* each of which represents an electrical circuit law. A constraint expression involves several cells—thus the voltage across a resistor, the current through it, and its resistance are related by a constraint expression which is an instance of Ohm's law for that particular resistor (see Figure 7).

When a model is made of a circuit diagram, a network of cells and constraint expressions is constructed. For example, consider the simple circuit in Figure 8. This circuit may be represented by the constraint diagram of Figure 9. (This is a simplification, SYN's constraint diagram is more complex.)

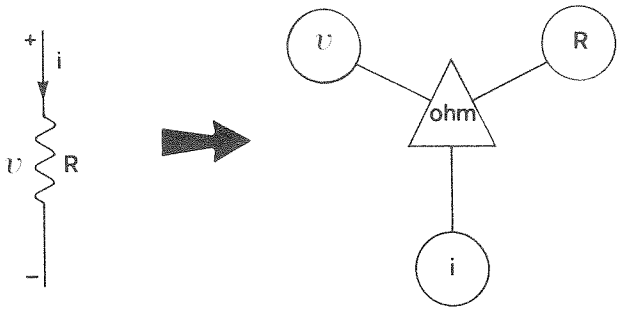

Figure 7. A resistor as a constraint diagram

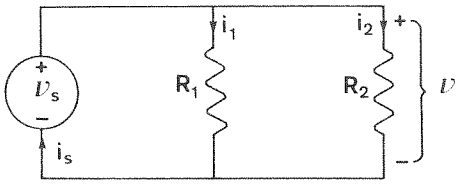

Figure 8. A simple circuit

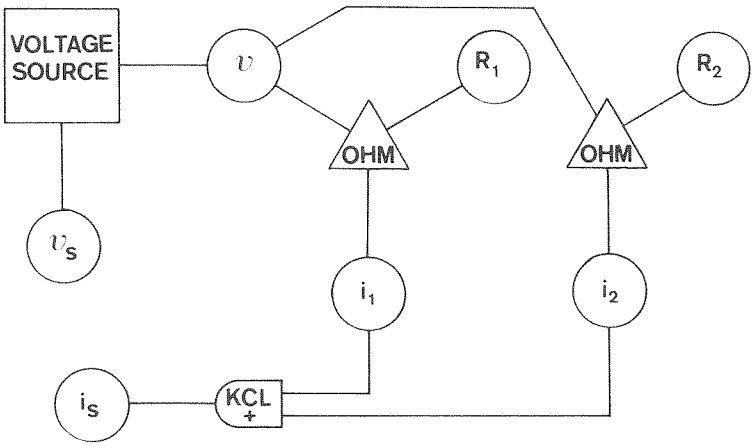

Figure 9. A constraint diagram for the circuit of Figure 8

Some of the cells in the constraint diagram for one model may be shared with the constraint diagrams of other, previously constructed models. For example, the resistance of a resistor is a cell shared among its ac and its dc models. So far, this network is just a way of representing the set of algebraic equations describing the circuit's behaviour. Propagation of constraints is a way of interpreting this network to combine the local constraints into global ones, thus analysing it.

Each cell may have a *value* (in SYN this is more complicated). The value may come from the user or it may be deduced from other values by constraint expressions. When a cell is assigned a value, each constraint it participates in is considered, to determine if enough information is available for it to be possible to use that constraint to deduce a value for another cell. Discovering a new value may thus determine many other values, thus 'propagating the constraints'. This could be accomplished in a parallel fashion, but in our machine, the parallelism is simulated with a queue.

Sometimes, two constraints can produce a value for the same cell. This condition is called a *coincidence*. If a coincidence occurs the values destined for the cell the coincidence occurs at must be the same for the set of constraints to be satisfiable. If the values are constant there are only two outcomes. If the constants are equal no new information is deduced, and if the constants differ, a contradiction has been found. A contradiction indicates that some faulty assumption has been made in that part of the design process not covered by this paper.[6] Sometimes, for symbolic analysis (and for overcoming simultaneity in the constraints), a value is a symbolic expression. In the case of a coincidence equating symbolic quantities, there is a third possible outcome. One symbolic quantity may be eliminated by solving for it in terms of the others. We will discuss the role of algebra later.

Consider an example: Suppose SYN knew the voltage of the source, V_s, the current into the source, I_s, and the resistance, R_1, in the circuit of Figure 8. Looking at the constraint diagram in Figure 9, we see that the only constraint which can make a deduction is the voltage-source law. Thus cell V is assigned a value equal to the value of cell V_s. The constraints attached to cell V are now examined to determine if any other deductions can be made. Ohm's law for the rightmost resistor cannot make a deduction because neither R_2 nor I_2 are known. But Ohm's law for the other resistor can combine the values of V and R_1 to produce a value for I_1. I_1 can be now combined with I_s (which was known originally) using KCL (Kirchhoff's current law) to produce a value for I_2. This value can now be used by Ohm's law to produce a value of R_2. This simple example illustrates the local similarity of analysis and synthesis. The problem of deducing the resistance is no different from the problem of deducing a voltage or a current.

The method does not always work so easily. Suppose that in the circuit of Figure 8 SYN was told R_2 instead of the voltage, V_s. No local constraint expression would have enough information to make any deductions by itself, though the behaviour of the network can be totally determined from the given information. The problem involves an inherent simultaneity in the constraints. This can be overcome by introducing a symbolic quantity and propagating it as if it were known. In this example, SYN could give cell I_1 the value α. Then it is possible to use KCL to deduce that I_2 is $I_s - \alpha$ and it is also possible to use Ohm's law to deduce that V is αR_1. These new values can be further propagated. Using Ohm's law on the other resistor allows us to determine that V is $(I_s - \alpha)R_2$. But SYN already knows a value for V. Hence there is a coincidence, thus the algebraic equation: $(I_s\text{-}\alpha)R_2 = \alpha R_1$ must be solved. It can be solved: $\alpha = (R_2/(R_1 + R_2))I_s$. The value of V is now known in terms of given parameters. That value can be propagated by the voltage-source law to give a value for V_s.

Examples like the second one illustrate the need for symbolic algebraic manipulation in a program which performs analysis by propagation of constraints. The simultaneity is apparent in the constraint diagram (see Figure 9) because there is a loop of constraints containing only unknown quantities. In the first example, the loop was broken when V was determined by propagation from V_s outside of the loop. In the second example, a symbolic unknown, α, was used to break the loop. The unknown could have been introduced anywhere in the loop, and the coincidence could have happened anywhere in the loop. The particular place is determined by the pseudo-parallelism of the queue. In essence, propagation is a means of constructing a small, dense set of equations from a large but sparse set. Propagation is applicable to non-linear constraints as well as linear ones (assuming that the algebraic manipulator which receives the equations can handle them). Unfortunately these examples are too small to see this phenomenon.

DEVICE MODELS

In this section we describe how devices are defined and modelled in SYN. We will see how parameters are specified and how they are shared among alternate models of the device. We will see how models may contain various hints on how they are to be used.

The first information that must be specified about a new device type is its set of terminals. These terminals are referred to by name inside the models of the device. They are also referred to by position when the device is wired into a larger circuit (as in the diagram construct used to make the cascode amplifier).

Thus we describe a transistor as a three terminal device with terminals named e, b, and c for emitter, base and collector:

```
(device transistor (e b c))
```

We now define several models of the transistor. For example, the bias characteristics of the transistor can be modelled in its active region by assuming that the base–emitter voltage of the transistor is a device parameter, and by assuming the collector current is a finite multiple, beta, of the base current. The bias model is described as follows. The model has a name, beta-finite, a list of applicable viewpoints, (dc), a list of parameters which may be shared with other models, (Vbe Beta Ic Polarity), and a body describing the constraints. In this model there are three constraints specified as equations. In addition, there are two hints we supply for the use of the variables. Some variables, for example Vbe, have been declared as opaque. This means that these variables should not be solved for unless forced to. This suggestion helps guide the choice of symbolic quantities which appear in the symbolic expressions. The current in the collector terminal of the transistor is declared to be key. This is a good place to put an unknown if it is necessary to break simultaneity.

```
(model transistor beta-finite (dc) (Vbe Beta Ic Polarity)
        (equation active ((v (voltage b e)))
                (- v (* Polarity Vbe)))
        (equation beta-finite ((Ib (current b)) (Ic (current c)))
                (- Ic (* Beta Ib)))
        (equation Ic-identity ((I (current c)))
                (- Ic I))
        (opaque Vbe Beta Polarity)
        (key (current c)))
```

The incremental model for a transistor illustrates an additional means of specifying a model. In this case we have a circuit model. The transistor is described as containing a wiring diagram similar to the ones produced by the diagram construct. The hybrid-pi model has one internal node and five parts connected among this node and the transistor terminals. Parameters of the internal devices, such as the capacitance of C_{pi} are identified with parameters of the transistor model. In addition, there are several equations also specified. The third equation, gm * r-pi = Beta, relates parameters from this incremental model to beta, a parameter from the bias model.

```
(model transistor hybrid-pi (ac)
        (c-mu c-pi r-pi r-x gm Ic Beta Ft Polarity)
        (wire (n)
                ((r-r-pi (resistor n e) (resistance r-pi))
                 (c-c-pi (capacitor n e) parasitic (capacitance c-pi))
                 (r-r-x (resistor b n) (resistance r-x))
                 (c-c-mu (capacitor n c) parasitic (capacitance c-mu))
                 (s-gm (voltage-controlled-current-source n e c e) (gm gm)))
```

```
(equation gm-Ic ()
        (- gm (* Ic q:k*T Polarity)))
(equation c-pi-cal ()
        (- c-pi (- (: gm (* two*pi Ft)) c-mu)))
(equation beta-gm-rpi ()
        (- (* gm r-pi) Beta))
(opaque c-mu c-pi r-pi r-x gm Beta Ft Polarity)
(key (voltage n e))))
```

FREQUENCY DOMAIN BEHAVIOUR

In the cascode synthesis we considered the midband model in order to avoid dealing with the capacitors. In the midband model, the coupling capacitors are short-circuited and the parasitic capacitors are open-circuited. SYN is also able to do s-plane analysis to determine the transfer functions of simple circuits. The difficulty is that the size of the symbolic expressions grows quickly with the size of the circuit. Even when it can fit in the computer, a three page expression is not very illuminating.

After finishing the midband and dc synthesis, all parameter values except for the coupling and bypass capacitances are numerically determined. (The parasitic capacitances in the transistors are determined by the dc bias conditions.) A complete ac analysis at this point would be somewhat more manageable because the numerical constants would combine and simplify. Unfortunately, the expressions are still too complicated to deal with easily. The incremental model of the cascode amplifier has seven poles! So the transfer function has a seventh degree polynomial in its denominator. This would be awful even if the coefficients were entirely numerical. One source of the complexity is that unnecessarily detailed models are used in the analysis. For example, r_x is often irrelevant; its presence considerably complicates the resulting expressions. If r_x were zero, at least one of those seven poles would go away. We currently have no idea for how to automate the choice of the simplest model appropriate for solving a problem.

Even when we can obtain the factored transfer function, we are constrained to specify the desired frequency response in terms of the poles and zeros of the circuit. But given the specification in terms of poles and zeros, we have no reason to believe that the algebraic solution for the capacitance values in the coefficient equations are tractable.

Specification in terms of poles and zeros is not very convenient for simple amplifiers. Designers usually specify the frequency response of an amplifier in terms of the dominant characteristics such as the 3 dB points. This type of specification leaves some degrees of freedom unspecified, to be further constrained by rules of thumb. For example, the designer usually tries to minimize the total capacitance. One common rule of thumb is to first determine the value required for each coupling and bypass capacitor to achieve the required low frequency 3 dB point (assuming all the other low frequency capacitors were infinite). The largest capacitor is then used at a slightly higher value. All the other capacitors are given ten times the value computed.

ALGEBRAIC MANIPULATION

SYN's algebraic manipulator is an adaptation of the rational canonical simplifier of MACSYMA. It reduces expressions built from a limited class of operators to a canonical form. The class of operators is addition, subtraction, multiplication, and division of symbolic quantities. It does not include transcendental functions. Although the complete description of the behaviour of a diode or a transistor involves exponential expressions, this limitation is not serious because SYN always deals with models of the behaviour of a device which avoid these exponential expressions. Rational canonical form is complete in its domain in that any two expressions that are algebraically equal have the same canonical form. In the canonical form of an expression, no symbolic quantity is mentioned which does not have a material effect on the value of that expression.

A rational expression is represented as a ratio of two relatively prime multivariate polynomials. Each polynomial is represented by a univariate polynomial in a selected variable with coefficients which are polynomials in the other variables. Each variable has unique global priority. No variable in a coefficient of a polynomial is of higher priority than the main variable of that polynomial. This guarantees a canonical form for polynomials. The fact that the polynomials of a rational expression are relatively prime guarantees a canonical form for rational expressions. For example, if the variables are ordered: C1 > C2 > C3 > R1 > R2 > R3, the following expression is in rational canonical form:

$$\frac{\text{C1 R1 R2 R3}+\text{C2 R1 R2 R3}+\text{C3 R1 R2 R3}}{\text{R1 (R2}+\text{R3)}+\text{R2 R3}}$$

Although the expressions that are generated by SYN are non-linear in device parameters, the algebraic manipulator is almost never forced to eliminate a variable which occurs in higher than first degree. In fact, a theorem due to Lin[7] ensures that if the only parameter variables in a linear, time-invariant network are ratios of voltages and currents (they may be impedances, admittances, voltage transfer ratios, etc.), then any network function may be expressed as the ratio of two polynomials of degree one in each variable parameter.

As Lin points out, this theorem is invalid of such parameters as a gyration resistance or a transformer turns ratio. More seriously, Lin's theorem is invalid if constraints on a parameter come from two circuits which share parameters (e.g. the dc and ac models of an amplifier), or from two *gedanken* experiments on the same circuit. Consider for example, the problem of determining the resistances in an L-network for matching impedances. Suppose we want the L-network shown in Figure 10 to match 75 Ω on the left and 50 Ω on the right:

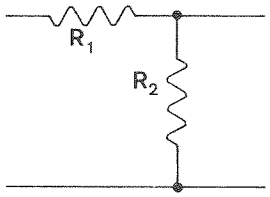

Figure 10. An L-network

We get constraints on the resistances from each of two gedanken experiments as follows:

$$(R_1+75)\|R_2=50$$

$$(R_2\|50)+R_1=75$$

These equations simplify to:

$$R_2=2R_1 \quad \text{and} \quad R_1R_2=3750$$

These equations are irreducibly quadratic; in fact we get:

$$R_1^2=1875 \quad \text{hence} \quad R_1=43{\cdot}3,\ R_2=86{\cdot}6$$

PROBLEMS WITH ALGEBRA

Unfortunately, SYN's ability to deal with large circuits is severely limited by practical difficulties. SYN spends most of its resources, both time and space, in algebraic manipulations. For circuits it can handle it is reasonably efficient. But the line between circuits it can handle and those it cannot is very sharp. For circuits

it fails on, it runs out of address space for storing temporary results during elimination of one variable. In fact, the final answer could easily fit.

One problem seems to be the use of rational canonical form. The canonical form is not necessarily the most compact form of the expression either for storage in a computer or for displaying for a user. For example, our canonicalized expression in resistance and capacitance (in the previous section) has 16 operators and multiple occurrences of the lower priority variables. A more succinct and natural representation of the same quantity is:

$$\frac{(C1+C2+C3)}{\frac{1}{R1}+\frac{1}{R2}+\frac{1}{R3}}$$

This form of the expression has only eight operators and no multiple occurrences of the variables. In this form it is obviously interpretable as the time constant of three resistors and three capacitors in parallel.

Although there is a unique canonical form relative to an ordering of the variables, there is a degree of freedom in the choice in the priorities of the variables. The apparent complexity of an expression can be very sensitive to the ordering. For example, if the variable ordering was chosen to be $R1 > R2 > R3 > C1 > C2 > C3$ this expression would only require ten operators to represent in rational canonical form:

$$\frac{R1\ R2\ R3\ (C1+C2+C3)}{R1\ (R2+R3)+R2\ R3}$$

When SYN is working on those circuits it eventually fails on, it is not uncommon to see expressions in seven or eight variables which take a couple of pages to print out. These expressions would be tractable if the ordering of the variables were different, but a global reordering would result in other simple expressions becoming prohibitively large. Perhaps we should abandon converting all expressions to a uniform canonical form. Instead, we should have a variety of possible representations for algebraic expressions and means for converting from one to another. We should then represent each expression in that form which is most economical.

When we examine how the algebraic manipulator spends its resources, we find that most of the effort (time and space) is used in computing the greatest common divisor (gcd) of the polynomials. In cases where SYN fails to complete a problem, it is always because just one gcd computation fills up the entire address space of the computer. In more detail, computation of a polynomial gcd involves computation of numerical gcd on the numerical coefficients. The actual limiting problem is that the sizes of the numerical coefficients in the intermediate results of the polynomial gcd get extremely large. New results in algebraic manipulation[8] may significantly help in this context.

There is probably no uniform answer to the algebra problem. Human engineers never do as complicated algebra as SYN's package can in fact do. In general, it is more helpful to have knowledge which tells us how to avoid manipulation than it is to have powerful manipulators. What additional knowledge do engineers use to guide the use of algebraic manipulation?

In order to avoid excessive algebraic manipulation, SYN uses electrical knowledge to guide the manipulations that are done. Propagation of constraints determines which expressions are considered at all. The other source of control SYN has over the algebraic manipulator is the variables it uses in the expressions. The choice of which variables to use, which to solve for, and which to substitute for can affect the amount of resources required to solve a circuit by as much as a factor of ten. The complexity of the algebraic expressions is very sensitive to which unknown quantities are used to express the others. Whenever propagation of constraints gets stuck it introduces a new algebraic variable to break a loop. SYN has considerable latitude to choose which cell in the loop receives the new symbolic quantity. The key declarations in the model definitions suggest good places to break loops. For example, the voltage across r-pi, if known, determines most of the other voltages and currents in a hybrid-pi model of a transistor.

When a coincidence results in an equation, SYN obtains a value for one of the unknowns in the equation. Although references to this variable now refer to its solved value, the expressions often remain much

simpler if the algebraic substitutions are not done. In the cascode example, SYN always substituted for solved values. SYN also has available a strategy which decides whether to substitute on the basis of the estimated complexity of the resulting expression. This complexity is measured in terms of the number of variables in the resulting expression. This strategy is computationally more efficient but the intermediate results generated are often obscure.

When solving an equation, SYN has a choice of which variable to eliminate. The engineer prefers to see his answers in terms of certain quantities. These preferred variables are called opaque. If possible, SYN will not eliminate an opaque variable. If applying this restriction does not result in a unique variable to solve for, SYN chooses to solve for that variable whose value would have minimal complexity.

CONCLUSIONS AND FUTURE WORK

Our experimental version of SYN is very weak, yet it is impressive. It has only begun to tap the power of algebraic manipulation in the synthesis of arbitrary circuits. We see several areas where extensive research is needed to make programs like SYN practical for use by engineers.

We need more ways to control the application of algebraic methods. The computer must know more about how to avoid doing algebra. Currently SYN uses only a few models and the human engineer controlling it must decide which model to use. Engineers use the simplest model which covers the relevant phenomena. We must develop programs which qualitatively understand circuits and can pick appropriate models. A related strategy that engineers use to simplify algebra is approximation. A good engineer knows which variables and terms to ignore in particular situations. Human engineers also make extensive use of terminal and port equivalence for focussing their attention. At any instant an expert will only concentrate on a small portion of a circuit, assuming that the rest will 'work as planned'. This assumption is captured by summarizing the behaviour of the circuit peripheral to the area under attention as a set of equivalences. Equivalences are also used to summarize the specifications of the behaviour of the part under attention. This allows the designer to reason about each section of the circuit in isolation.

Another dimension of improvement is the introduction of more powerful ways of describing the requirements we have for a circuit. Many requirements are convenient to specify by inequalities. For example, the fact that we want our amplifier to support an output swing of 1 V, or that we want it to have an input impedance greater than 47,000 Ω. Unfortunately the algebra of inequalities is almost intractable and a great deal of research will be required to include such specifications. Another type of specification we must be able to assert is the frequency-domain behaviour of a circuit. For example, we would like to be able to say that an amplifier is flat from 200 Hz to 10,000 kHz and then make use of that requirement in synthesis. We are still far from effective techniques for stating and using such requirements.

Finally, we would like to have the synthesis program know reasonable rules of thumb which it uses as a default unless the user specifies a contradictory requirement. This is in itself not hard, but the program must then understand the reasons behind the rules of thumb in order to know which to release when a contradiction is encountered. The rules of thumb often depend upon understanding simpler circuits and how compound circuits are built out of the simpler ones. For example, the bias rules for the cascode amplifier are really bias rules for the common emitter stage in it. We believe that it is feasible to have a program which not only synthesizes device parameters in circuits but which actually pastes together subcircuits to make compound circuits like cascode. We are currently engaged in such research.

Even if these difficulties were dealt with, the ideal capabilities of SYN constitute only a fragment of those of a human designer. In particular, the construction of plausible circuit topologies from descriptions of the overall purpose of the circuit is not addressed. Our research on this subject thus far suggests three directions which need to be explored. First, circuits are designed hierarchically by describing the circuit at a function block level before considering the specific topologies of the blocks. Second, engineers utilize a library of stereotypical circuit fragments which they patch together to form plausible circuit topologies. Third, engineers often construct qualitative models of the mechanisms which manifest the desired input-output behaviour before considering possible circuit topologies.

NOTES AND RELATION TO OTHER WORK

1. There are other approaches to the automation of the process of synthesis of general circuits. One approach uses numerical optimization of the unknown parameters relative to a given objective function. Though the method depends upon having a network of linear elements, it is possible to extend this method to synthesis of nonlinear circuits by iterating this process on models linearized to an operating point [TABLEAU]. There are many specialized programs which synthesize particular classes of topologies, especially for filters made up of ladder or lattice structures. For example, [Ladder] describes algorithms for optimal synthesis of a small class of passive bandpass filters. Modern specialized synthesis techniques can even generate new topologies for very specific situations. For example [Volt Mult] explains how to generate and analyse novel voltage multiplier circuits.

2. Most computer-aided circuit analysis programs are purely numerical. These have been used as a basis for user-oriented systems of great versatility and convenience, for example [MINNIE] augments numerical circuit analysis methods with impressive graphics. On the other hand, many people have realized that considerable insight can be gained by examining the symbolic expressions for circuit behaviour. [Survey] surveys the applications of symbolic network functions in which some network element parameters, in addition to the complex frequency, are represented as symbolic variables. One efficient means of constructing the symbolic network function is by interpolation for the coefficients of the network function polynomials [Interp], this can even be done in the light of symbolic parameter values, though with less efficiency [Graph]. In [NAPPE], the method of parameter extraction is used. This is much more like the symbolic algebraic manipulation which we use, though it still depends upon numerical techniques which may lose precision and which may be untraceable.

3. Propagation of constraints is a method of circuit analysis which was derived by generalizing Guillemin's informal method for solving ladder networks. Propagation analysis uses one-step deductions to derive as much as possible about a circuit without the introduction of any algebraic unknowns. If there remain values to be determined, a symbolic quantity is postulated as the value of an unknown and is further propagated. This process results in the construction of a small number of algebraic equations. Propagation is described in greater detail in [WATSON], [EL], [INTER], [ARS]. The best description of propagation analysis can be found in [ARS]. The application of propagation to synthetic reasoning is discussed in [SLICES].

4. As described in the section 'Problems with algebra' we have adapted the rational canonical simplifier of MACSYMA [MACSYMA], a powerful algebraic manipulation system developed by the MIT Mathlab group led by Joel Moses.

5. The small-signal transconductance of a transistor is proportional to the collector bias current: gm = (q/kT)Ic. The parameters of a transistor are also related in other ways, for example: gm Rpi = beta.

6. This work is part of a larger effort to apply artificial intelligence methods to computer-aided design at the MIT AI Laboratory. An overview of the work is in [Overview]. Drew McDermott [NASL] has made considerable progress on the problems of rephrasing problem descriptions and proposing plausible designs. Allen Brown [WATSON] and Johan de Kleer [INTER] have investigated the problems involved in troubleshooting circuits—with and without access to a plan. Richard Stallman and Gerald Sussman have designed and implemented some novel circuit analysis programs [EL] [ARS]. Johan de Kleer [JdK Prop] [Thesis] is considering the problem of qualitative causal analysis of a circuit. We are not solely interested in computer-aided circuit design, but also in an understanding of the general epistemology of engineering [Prog Eng].

7. This theorem is stated and proved in a beautiful paper by P. M. Lin [Survey]. We had noticed that SYN was never forced to eliminate variables of higher than first degree and were terribly confused by this for some time. Lin's theorem helped considerably and will probably be quite useful in our improvements to our algebraic manipulator.

8. Richard Zippel (in a forthcoming PhD thesis at MIT) has recently developed algorithms for combating this 'intermediate expression explosion' in the case of polynomial GCD where the answer is much smaller than the input expressions. His idea is a breakthrough on this problem and will probably make many previously infeasible problems tractable.

REFERENCES

[ARS] Richard Stallman and Gerald Jay Sussman, 'Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis', *MIT Artificial Intelligence Laboratory Memo 380*, Cambridge, Massachusetts, (1976). Also in *Artificial Intelligence* **9**, 135–196 (1977).

[EL] Gerald Jay Sussman and Richard Stallman, 'Heuristic techniques in computer-aided circuit analysis', *IEEE Trans. Circuits and Systems*, **CAS-22**, 857–865 (1975).

[Graph] Kishore Singhal and Jiri Vlach, 'Symbolic analysis of analog and digital circuits', *IEEE Trans. Circuits and Systems*, **CAS-24**, 598–609 (1977).

[INTER] Johan de Kleer, 'Local methods for localization of faults in electronic circuits', *MIT Artificial Intelligence Laboratory Memo 394*, Cambridge, Massachusetts, (1976).

[Interp] J. K. Fidler and J. I. Sewell, 'Symbolic analysis for computer-aided circuit design—the interpolative approach,' *IEEE Trans. Circuit Theory*, **CT-20**, 738–741 (1973).

[JdK Prop] Johan de Kleer, 'A theory of plans for electronic circuits', *MIT Artificial Intelligence Laboratory Working Paper 144*, Cambridge, Massachusetts, (1977).

[Ladder] Herman Daae, 'Computer-aided optimal synthesis of bandpass filters', *IEEE Trans. Circuits and Systems*, **CAS-22**, 490–495 (1975).

[MACSYMA] The Mathlab Group, *MACSYMA Reference Manual*, MIT Laboratory For Computer Science, Cambridge, Massachusetts, (1977).

[MINNIE] Robert Spence and Mark Apperley, 'The interactive-graphic man–computer dialogue in computer-aided circuit design', *IEEE Trans. Circuits and Systems*, **CAS-24**, 49–61 (1977).

[NAPPE] P. M. Lin and G. E. Alderson, 'Computer generation of symbolic network functions—a new theory and implementation', *IEEE Trans. Circuit Theory*, **CT-20**, 48–56 (1973).

[NASL] Drew McDermott, 'Flexibility and efficiency in a computer program for designing circuits', *MIT Artificial Intelligence Laboratory Technical Report 402*, Cambridge, Massachusetts, (1976).

[Overview] Gerald Jay Sussman, 'Electrical design—a problem for artificial intelligence research', *MIT Artificial Intelligence Memo 425*. Also in *Proc. IJCAI V.*, (August 1977).

[Prog Eng] C. Rich, H. Shrobe, R. Waters, G. J. Sussman and C. Hewitt, 'Programming viewed as an engineering activity', *MIT Artificial Intelligence Laboratory Memo 459*, Cambridge, Massachusetts, (1978).

[SLICES] Gerald Jay Sussman, 'SLICES: at the boundary between analysis and synthesis', *MIT Artificial Intelligence Laboratory Memo 433*, Cambridge, Massachusetts, (1977).

[SNAP] P. M. Lin and G. E. Alderson, 'SNAP—a computer program for generating symbolic network functions', *Technical Report TR*–EE70–16 School of Electrical Engineering, Purdue University, Lafayette, Indiana, (1970).

[Survey] P. M. Lin, 'A survey of applications of symbolic network functions'. *IEEE Trans. Circuit Theory*, **CT-20**, 732–737 (1973).

[TABLEAU] Gary D. Hachtel, Robert K. Brayton and Fred G. Gustavson, 'The sparse tableau approach to network analysis and design', *IEEE Trans. Circuit Theory*, **CT-18**, 101–113 (1971).

[Thesis] Johan de Kleer, 'Causal and teleological reasoning in circuit recognition', *MIT Artificial Intelligence Laboratory Technical Report 529*, Cambridge, Massachusetts, (1979).

[Volt Mult] P. M. Lin and Leon O. Chua, 'Topological generation and analysis of voltage multiplier circuits', *IEEE Trans. Circuits and Systems*, **CAS-24**, 517–530 (1977).

[WATSON] Allen L. Brown, 'Qualitative knowledge, causal reasoning, and the localization of failures', *MIT Artificial Intelligence Laboratory Technical Report 362*, Cambridge, Massachusetts (1975).