In: ACM SIG SIM Simulation Digest. Vol. 19, Iss. 4, Winter 1988/1989. pp. 59-73

A Knowledge-Based Simulator for Electronic Circuits

Jürgen Herczeg, Michael Herczeg

Research Group INFORM
Computer Science Department
University of Stuttgart
Herdweg 51
D-7000 Stuttgart 1
Federal Republic of Germany

Abstract

The following paper describes a prototypical knowledge-based simulator for electronic circuits supporting the user during various phases of the task. The system has been built by augmenting an object-oriented modelled simulator by direct manipulation techniques and by representing knowledge about the simulation process and the application domain in an expert component. The direct manipulation interface enables the user to manipulate the familiar objects of an electronics laboratory via a mouse on the screen. The expert component provides adequate simulation parameters, answers questions about the application domain, and gives examples and partial solutions concerning the user's current problem.

Keywords: knowledge-based simulation, electronic simulator, direct manipulation, knowledge representation, expert system, object-oriented programming

1 Introduction

Important progress has been made in the design of simulation systems when the early batch-oriented simulators have been replaced by interactive systems providing a more appropriate way of specifying simulation input data and displaying simulation results.

However, some of the problems have still remained. Usually the user has to specify various simulation parameters depending on the simulation model, the input data, and the needed quality of the results. The desired visualization of the results has to be chosen and adjusted depending on the range and domain of the result values. Perhaps the most difficult problem remaining, is to decide how the input data have to be changed to improve the results according to the goals of the

simulation. A major shortcoming is that the systems cannot be asked for explanations concerning the simulation environment and the application domain [Rothenberg 88].

Several problematic tasks when using a simulation system can be isolated:

Data Specification:	often large amounts of complex input data have to be fed into the system
Simulator Configuration:	the simulator has to be configured to produce meaningful results
Result Evaluation:	the results of the simulation have to be presented in an adequate way to be interpretable by the user
Modification:	the input data and/or the simulator configuration have to be modified in order to get new improved results

What is needed, is a kind of intelligent support rendered by the simulation system. To demonstrate such support features in a prototypical system, we have built the electronic simulator ELAB, which has been restricted to four-poles, a special class of electronic networks. To support the specification of input data as well as the interpretation of simulation results, we have built a graphics-based direct manipulation interface on top of the simulator. To cope with the problems of the simulator configuration before a simulation as well as the problems of modification after a simulation we augmented the direct manipulation laboratory ELAB by the expert component ELEX. ELEX automatically configures the simulator according to the model and the input data. It actively warns the user when problems may arise and passively (i.e. on demand) gives advice about application concepts, simulation results, and their interdependencies. For these purposes, ELEX has several knowledge sources represented in object-oriented and rule-based paradigms.

The system has been implemented in the object-oriented, LISP-based knowledge representation language ObjTalk [Lemke 85] on a VAX 11/780 with high resolution bitmap terminals. We were able to use software tools, like an existing user interface toolkit based on the window system environment WLISP of the research group INFORM [Böcker, Fabian, Lemke 85].

2 ELAB - A Direct Manipulation Simulator

With ELAB, the analysis of electronic circuits may be performed on a computer in essentially the same way as in a real laboratory. The user sees an experimentation field with most of the familiar working objects of an electronics laboratory (Figure 1):

- electric devices (resistors, capacitors, coils) represented as elementary four-poles
- voltage source (function generator)

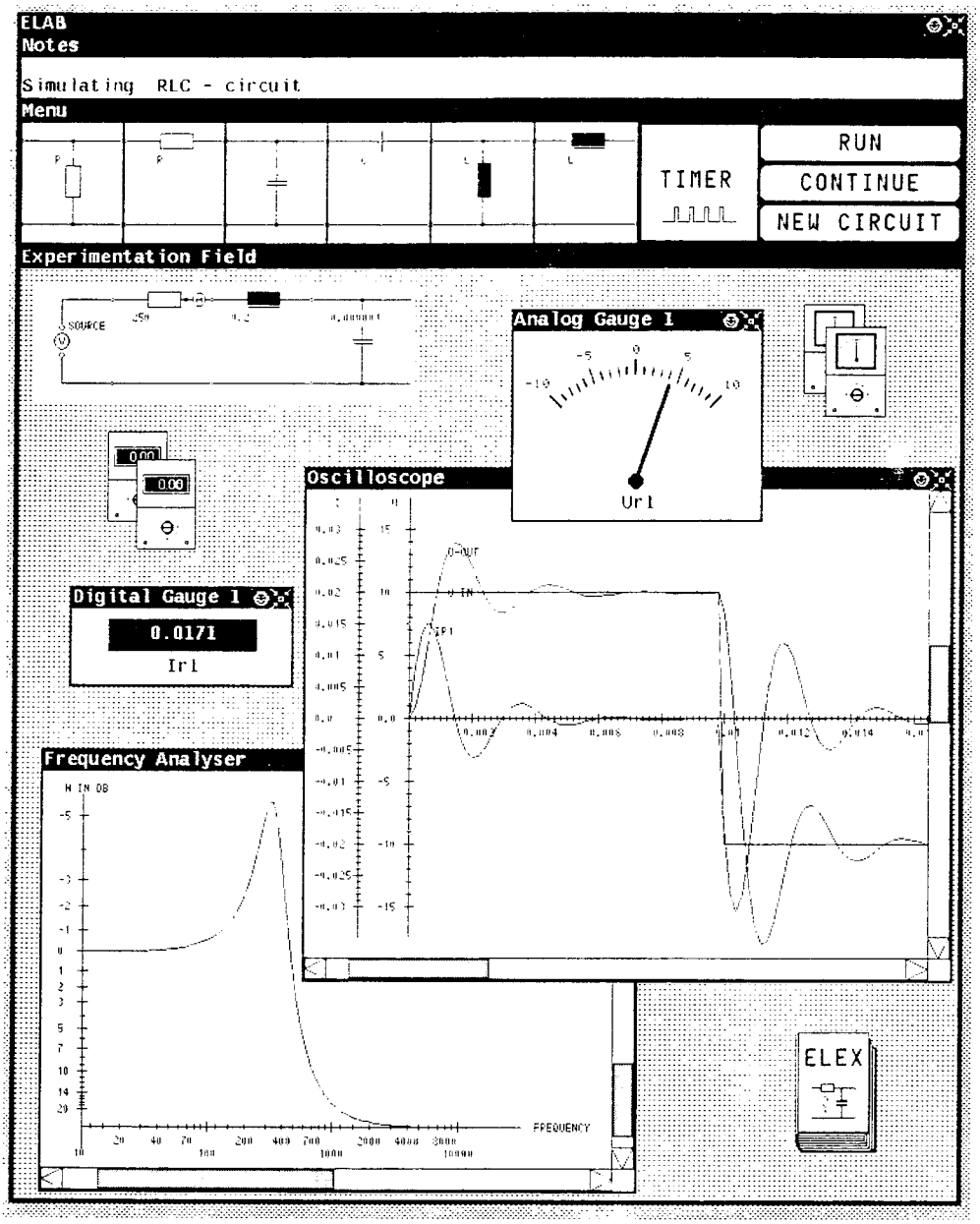

Figure 1. The Experimentation Field of ELAB

- measuring instruments (oscilloscope, frequency analyser, analog and digital gauges)
- electronics handbook (the dynamic visualization of the electronics expert)

These objects are manipulated by means of *direct manipulation* [Shneiderman 83; Hutchins et al. 86]. A circuit is built by connecting *elementary four-poles* to the source. This is performed with the mouse as a pointing device. Various measuring instruments may be connected to the circuit. *Pop-up-sheets* are used to

change the attributes of the four-poles and the instrument settings. By pressing a *softbutton* the circuit may then be "run" by the simulation component. The user can modify the circuit until it shows a satisfying behavior. To accomplish this task it is not necessary to learn a formal language (like in the SPICE system [Vladimirescu, Newton, Pederson 80]) or to use a complex computer-aided design system or a graphics package. The user solely learns to select, move, connect, and modify objects on the screen. The desktop metaphor has been transformed into a *lab metaphor*: the visualization of a laboratory on a computer screen.

ELAB is restricted to a small application domain, a microworld: The system allows the combination of elementary four-poles consisting of *resistors*, *capacitors* or *coils* into more complex circuits. However, the resulting circuits, e.g. *voltage dividers*, *filters* and *resonant circuits*, can get so complex that even a human expert is unable to predict their behavior qualitatively, let alone quantitatively. Diverse input signals, such as constant signals, pulses, and periodical signals in the shape of sine, rectangle, or triangle, and initial states of the devices, e.g. charged capacitors, increase the complexity drastically.

Various instruments are used to analyze the circuit behavior. To present different aspects of a circuit they employ different kinds of simulations:

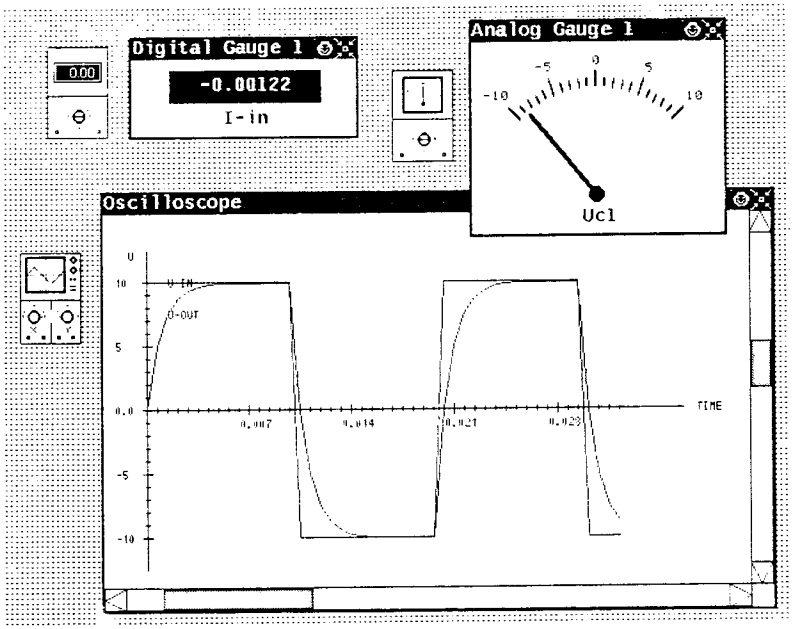

Figure 2. Analog Gauges, Digital Gauges and Oscilloscopes

Probably most familiar is the so-called *transient analysis* based on simulation in the *time domain*. The results of this kind of simulation are presented to the user via *analog* or *digital gauges*, showing momentary values, and *oscilloscopes*, displaying the results sampled over the simulation time (Figure 2). They look like hardware instruments of a real laboratory, but have some significant advantages. The oscilloscope, for example, is able to protocol simultaneously as many different signals as the user wants to see. Its coordinate system is variable and it displays not only periodical signals like most hardware oscilloscopes do, but also arbitrarily short pulses and varying oscillations. The analog gauges have interactively modifiable scales, hands and labels.

Very useful to analyze the behavior of a specific class of four-poles, called filters (e.g. low-passes, high-passes, bandpasses), is the *frequency domain analysis*. For this end ELAB provides a *frequency analyser*, an instrument that displays the *transfer function* of a circuit, i.e. the input/output signal ratio, in an arbitrary frequency interval (Figure 3).

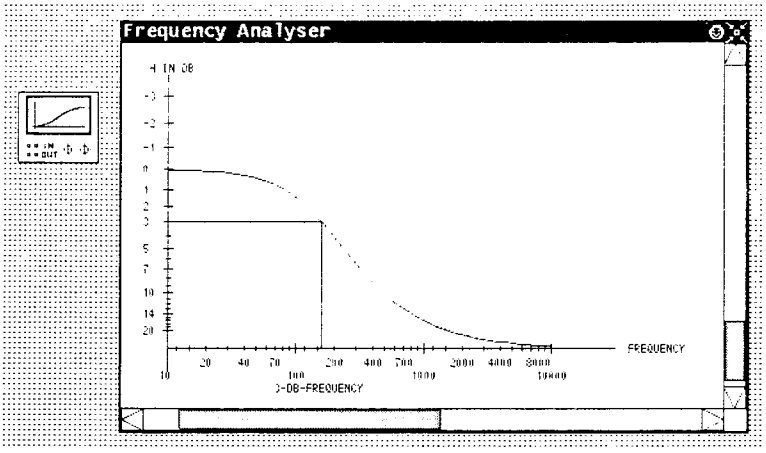

Figure 3. Frequency Analyser

An additional instrument of ELAB usually not found in electronic simulation systems, though very helpful, is the *electronics expert* allowing an *expert analysis* of a circuit on a more abstract level. It provides textual information on the present circuit and quantitative as well as qualitative information about characteristic circuit parameters (Figure 4).

The electronics expert is one part of an expert component integrated into the ELAB system, called ELEX, which combines aspects of an intelligent tutoring system [Sleeman, Brown 82] and an expert system.

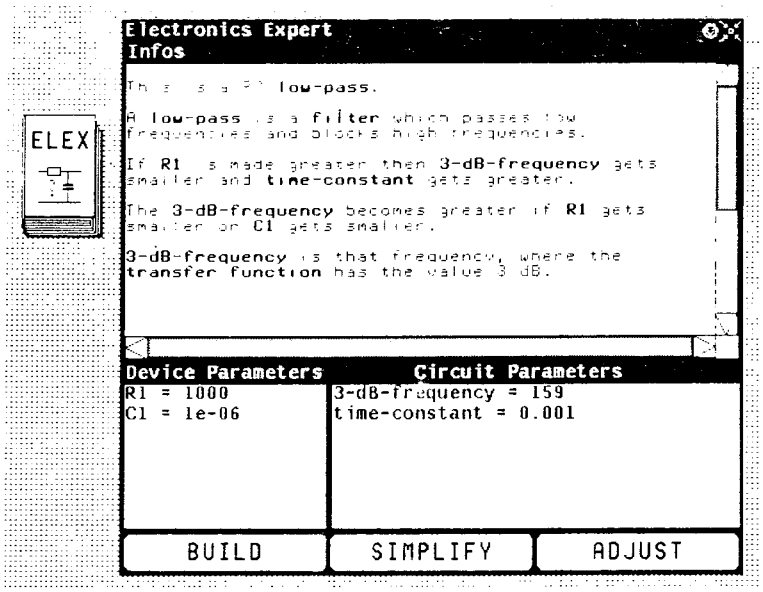

Figure 4. Electronics Expert

ELEX

- "watches" and criticizes the user's actions;
- controls simulation parameters;
- adjusts the displays of the measuring instruments in order to provide a good overview of the sampled information;
- simplifies complicated circuits to more compact ones with the same behavior by substituting and removing elementary four-poles;
- builds a particular circuit specified by the user according to its qualitative classification and not its structural components (e.g. the user specifies that a *RC low-pass* shall be built);
- describes the present circuit and explains related electronics concepts, enabling the user to browse through a network of concepts (e.g. the user requests an explanation of the concept *low-pass* or the parameter *3dB-frequency*);
- gives statements about the qualitative interdependencies between the device parameters and the characteristic circuit parameters, e.g. how the resistance *R1* of the present circuit affects the circuit parameters, or how the *3dB-frequency* is affected by the device parameters;
- adjusts the parameters of the devices according to a high level description specified by the user (e.g. the user wants a *resonant circuit* to have a given *resonant frequency*);

Experience with earlier versions of the ELAB system without ELEX showed how important such an expert component is, even for systems that may be handled easily, like direct manipulation systems, if there is a complex application domain. Thus, ELEX turned out to be a key component for a knowledgeable experimentation environment.

3 The System Components

ELAB has been implemented in the object-oriented language ObjTalk. Nearly every object shown on the display is represented as an internal object. The third elementary four-pole of the circuit in figure 5, for example, is represented by an internal object with the following attributes:

```
Type = resistor              the four-pole consists of a resistor
Orientation = in_series      the resistor is arranged in series
Quantity = R                 the characteristic quantity is resistance R
R = 1000                     resistance in Ohms
U = 10                       voltage at the resistor in Volts
I = 0.01                     current through the resistor in Amperes
Position = 3                 the four-pole is the 3rd four-pole of the circuit
Relative-Position = 2        the four-pole is the 2nd resistor of the circuit
Circuit = <some-circuit>     the circuit which the four-pole belongs to
Icon = <some-object>         icon to visualize the four-pole
Sheet = <some-sheet>         sheet to modify object attributes of the four-pole
U-to-be-Checked? = t         voltage at the resistor is checked by the
                             oscilloscope
I-to-be-Checked? = nil       current through the resistor is not checked
```

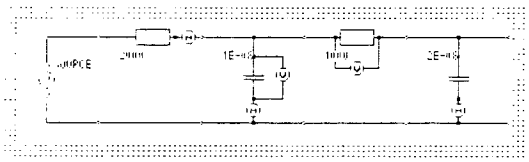

Figure 5. A Circuit composed of four Elementary Four-poles

Besides its attributes, an object has methods that represent its behavior. An instrument object, for example, has methods to connect or disconnect itself to or from an elementary four-pole object of the circuit.

ELAB is composed of seven functional components (Figure 6). These components are only conceptually defined, not all of them are reflected in the implementation. Every component consists of a set of objects. The lines in Figure 6 show the communication paths between objects of different components. The reason for identifying components is to group together objects which perform some particular task in the system. The whole system forms a large network of communicating objects.

The functional components of ELAB are:

Circuit:	object representing the circuit constructed by the user
Voltage Source:	object that drives the *Circuit* by a user selected input signal
Measuring Instruments:	objects that may be connected to the *Circuit* to analyze its behavior

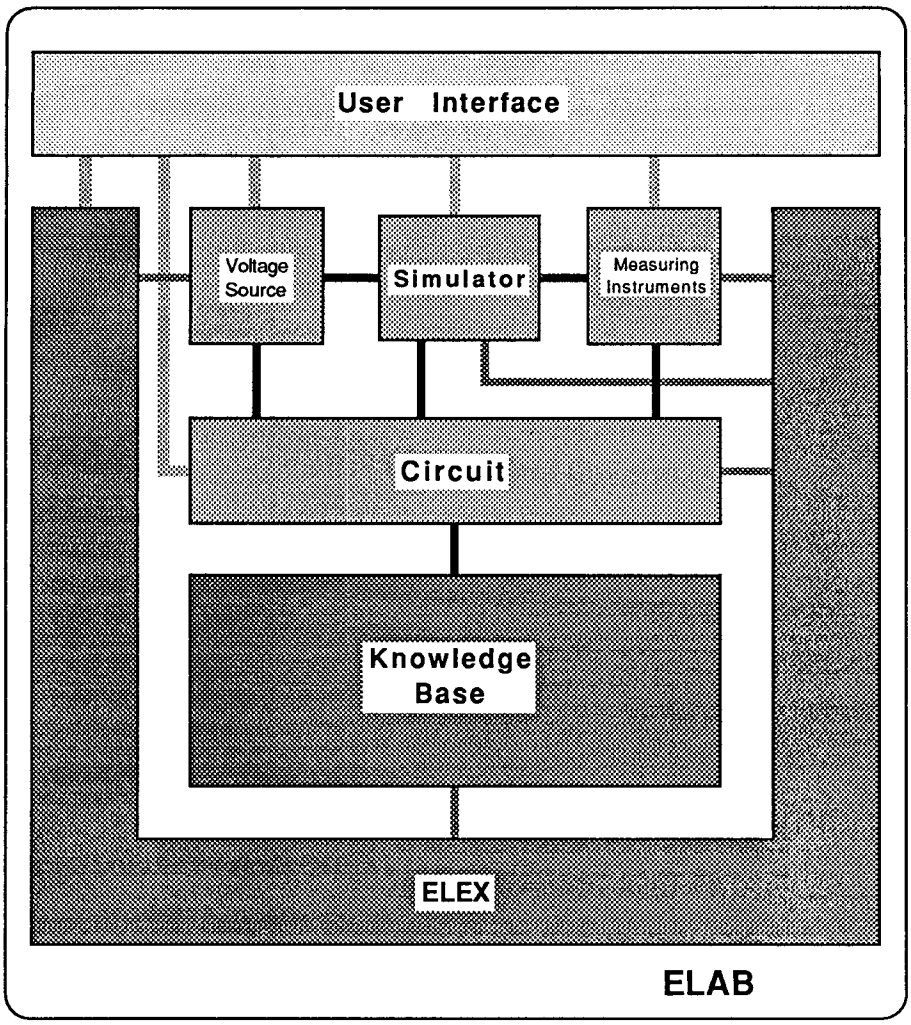

Figure 6. The Components of ELAB

Simulator: a timer generating "time-ticks" to make *Voltage Source*, *Circuit*, and *Measuring Instruments* proceed in time and produce the simulation data for the transient analysis

Knowledge Base: objects representing electronics concepts and rules to create the *Circuit* according to the user's graphical specifications, to provide the simulation knowledge for each circuit, and to generate explanations requested by the user via ELEX

ELEX: objects that represents the expert component; ELEX communicates with all other components, it can functionally be divided into three major parts:

- a *Simulation Expert*, managing the access to the circuit objects in the *Knowledge Base* and setting up the simulation parameters,

- a *Laboratory Expert*, serving as a tutor that watches the user at work with the ELAB environment, and
- an on-call *Electronics Expert*, providing different kinds of information about the application domain.

User Interface: user interface objects displaying the internal objects of the other components and processing input events invoked via mouse and keyboard

4 The Simulation and Expert Components

ELAB is based on simulation techniques combined with knowledge representation, retrieval and inference methods. The system components performing these tasks are the *Simulator*, the *Knowledge Base*, and ELEX.

4.1 The Simulation Model

A circuit to be simulated in ELAB is specified by a sequence of elementary four-poles. The *Simulation Expert* determines the corresponding object from the *Knowledge Base* describing the simulation behavior of the circuit. This means that only those circuits can be simulated which can be mapped onto a circuit object of in the *Knowledge Base*. The *Knowledge Base*, of course, can be extended by adding the description of any circuit that can be built from the elementary four-poles.

The simulation model for thetransient analysis is based on a numerical process that utilizes the so-called *step function response* of the circuit, i.e. the circuit's response to a constant input signal. This response is completely specified by the change of those electric quantities which determine the energy in the circuit, i.e. the capacitor voltages and coil currents, and by the dependencies of all other quantities (device voltages and currents) on those energy-determining quantities. The step function response of the circuit depending on the constant value of the input voltage, the device parameters, and the state of the circuit, i.e. the previous values of the energy-determining quantities, is represented in the circuit object. To get the overall behavior of a circuit driven by an arbitrary input signal the process of evaluating the step function response is iterated on short constant time intervals. The constant input voltage for each step of the iteration is obtained from the value of the input signal scanned at the beginning of each time interval. After each step, the newly calculated values for voltages and currents are propagated to the measuring instruments as required. The whole simulation process is controlled by one central object which synchronizes the communication between the source, the circuit, and the measuring instruments. The numerical error of this procedure is held low if the length of the constant time intervals is

small. On the other hand, the number of iterations to be performed in order to reach a given total simulation time increases with shorter time intervals. A trade-off between these simulation parameters is made by the *Simulation Expert*.

The simulation data presented by the frequency analyser and the *Electronics Expert* are derived from the dependencies of the transfer function and the characteristic circuit parameters on the parameters (resistance, capacitance, inductance) of the electric devices. These interdependencies are explictly represented in the circuit objects of the *Knowledge Base*.

4.2 The Knowledge Base

The simulation and expert components utilize different kinds of knowledge. While procedural knowledge is implicitly included in the simulation and knowledge inference procedures there is a fair amount of declarative knowledge explicitly represented in the *Knowledge Base*.

The *Knowledge Base* consists of a collection of circuit objects, a hierarchy of higher level circuit concepts, and a set of arbitrary electronics concept descriptions. Additionally, there are rules that describe how a circuit specified by a sequence of elementary four-poles can be transformed or possibly simplified to a definite internal form.

Each circuit object represents a circuit with a particular behavior that can be built out of elementary four-poles in possibly different ways. Besides the *normalized four-pole structure* by which it is selected from the *Knowledge Base*, a circuit object defines the step function response for the transient analysis, the transfer function applied by the frequency analyser, and characteristic circuit parameters presented by the *Electronics Expert*. Additionally, there is a pointer to the next general concept of the circuit concept hierarchy and, optionally, a brief textual description of the circuit.

Concepts are represented as objects with a *description-part* and a *feedback-part*. The *description-part* holds a text that explains the concept. Since this text may refer to other concepts, many concepts may be connected implicitly. The *feedback-part* describes optional visual feedback actions for a better explanation of the concept, e.g. when the concept *3dB-frequency* is explained its value in the present circuit is drawn into the diagram of the transfer function (Figure 3).

Circuit concepts are explicitly connected by pointers (*sub-circuit* pointers and *super-circuit* pointers) to represent the natural concept-subconcept relation. The concept *low-pass*, for instance, has a *super-circuit* pointer to the concept *filter*, whereas *filter*, on the other hand, has a back-pointing *sub-circuit* pointer to *low-pass*. Each circuit concept may have a set of associated circuit parameters which

characterize the concept precisely; for example, *low-pass* has the associated parameter *3dB-frequency*. The terminal nodes of this hierarchical circuit network are the circuit objects described above. They have a pointer to their immediate *super-circuits* in the hierarchy, e.g. *RC low-pass* has a *super-circuit* pointer to *low-pass*.

The *Knowledge Base* may be extended easily by adding new circuit objects or concepts without changing the simulation or expert component.

4.3 The Expert Component ELEX

ELEX consists of three subcomponents addressing different tasks in the system: the *Simulation Expert*, the *Laboratory Expert* and the *Electronics Expert*.

The Simulation Expert

By a pattern-matching approach the *Simulation Expert* determines the circuit object from the *Knowledge Base* that represents the present circuit. It transforms the sequence of elementary four-poles into a normalized form by internally reordering the four-poles according to a set of precedence rules represented in the *Knowledge Base* that maintain the behavior of the circuit. The corresponding object from the *Knowledge Base* is found by matching this normalized form with the *normalized four-pole structure* represented in the circuit objects.

There are a few simulation parameters to be adjusted for each simulation process of the transient analysis, such as the constant time interval length for each iteration and the number of iterations to be performed in order to get a usable simulation result. Since the ELAB user is usually not concerned with the simulation model and does not know the behavior of the circuit in advance, he is not able to adjust these parameters properly. Therefore, he is relieved from this work by the *Simulation Expert*, which automatically optimizes the simulation parameters for each simulation run, utilizing knowledge about the input signal, the circuit behavior, and possible requirements on the quality of the simulation stated by the user.

The Laboratory Expert

The *Laboratory Expert* supports the ELAB user while setting up experiments. It ensures that the system is always in a consistent and reliable state. Moreover, it directs the attention of the user to critical situations that are obviously not intended. Since in most cases these stem from ignorance or negligence on the part of the user, they can only be overcome by an active component. The *Laboratory Expert* watches the interactions and makes the user aware of mistakes, such as wrong parameter values entered by the user, and of critical situations, e.g. a

simulation run without any connected measuring instruments. While mistakes are simply undone, in critical situations the expert gives the user the opportunity to undo the action that was probably not intended. To detect these kinds of problems, continuous communication is required between the expert and the experimentation objects, e.g. the measuring instruments. The expert applies heuristics to decide whether and when the actions of the user are to be interrupted. This is a characteristic problem of active help systems and intelligent tutoring systems.

The Electronics Expert

The *Electronics Expert* supports the user in building and understanding circuits and enables the user to solve problems by communication on a more abstract level of the application domain.

The dialog between the user and the expert takes place in a window containing three subwindows - one for textual information, another for the parameters of the electric devices (*resistance R*, *capacitance C* and *inductance L*), and a third for characteristic parameters of the circuit, e.g. the *3dB–frequency* of a *low–pass filter* - and some softbuttons to trigger specific actions of the expert (Figure 4). Each piece of information has to be requested by the user, i.e. the expert does not present all available information at once; it only shows what may be particularly interesting for the user. Actually, information is partly generated at the moment it is requested, by making inferences from general knowledge and specific circuit data.

When a new circuit is built, there are an initial description and the parameters of the circuit combined with their present values. All characteristic circuit parameters are found starting at the circuit object in question following the *super–circuit* path through the net. This corresponds to the concept of inheritance in hierarchical networks. To request information about concepts, the user points to the item in the expert window in which he is interested. Such sensitive items are printed in bold face and are explained when selected by the mouse, i.e. the *description–part* of the concept is printed in the information window, thus possibly providing more concept items to be selected, and the *feedback–part* is executed. This is closely related to the current *hypertext* approaches, where textual and pictorial information is tightly interwoven [Smith, Weiss 88].

The *Electronics Expert* can also build special circuits according to their qualitative classification. Instead of connecting elementary four-poles the circuit is specified by refining circuit concepts by *sub–circuits* chosen from a menu, e.g. a *RC low–pass* is specified by the circuit concept path *four–pole - filter - low–pass - RC low–pass*. The specified circuit is built by the expert. Conversely, circuits built by the user can in some cases be simplified according to the internal representation of a circuit object from the *Knowledge Base* having the same overall behavior,

e.g. two resistors in parallel can be substituted by one resistor. Those simplifications can be determined from the *normalized four-pole structure* of the circuit by simplification rules represented in the *Knowledge Base*.

Explanations of interdependencies between parameters may be requested by selecting a parameter and choosing some query from a menu (Figure 7). This qualitative information can be particularly useful to understand how a circuit works, and how its behavior is influenced by the parameters of its components [Rothenberg 88]. Qualitative interdependencies between device parameters and circuit parameters are derived by numerical evaluation. To find out how the parameter of a particular device affects a specific circuit parameter, the *Electronics Expert* conducts an experiment: it changes the value for the device parameter, watches the effect on the circuit parameter, and determines the qualitative dependency. The user requesting the information does not notice this experiment conducted internally. The result is presented in a textual form.

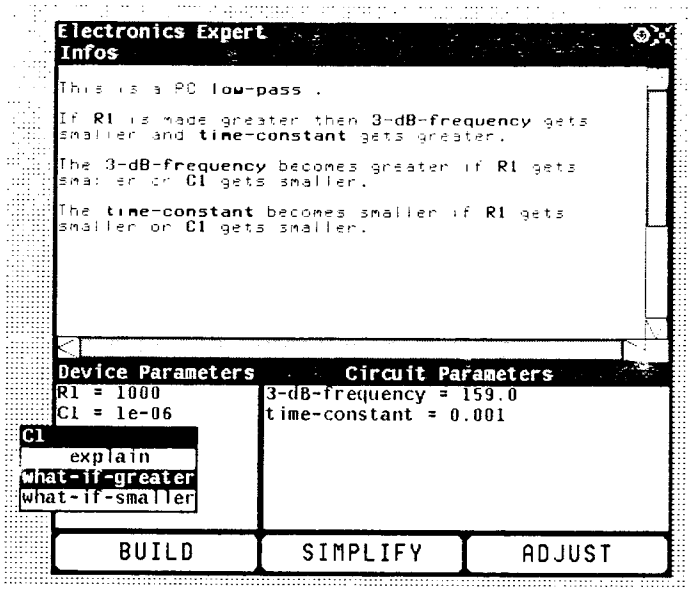

Figure 7. Asking for Parameter Dependencies

The *Electronics Expert* also helps the user to adjust the device parameters so that the circuit shows a particular behavior, e.g. it is able to set the *3dB-frequency* of a *low-pass* to 4 kHz. To get a particular circuit behavior, a circuit parameter is selected and certain parameter constraints to be satisfied by the expert may be specified in a special sheet (Figure 8). To solve this constraint problem a *relaxation* method based on a numerical, iterative process is employed that determines the roots of an equation with several variables [Jayaraman 84]. The equation is derived from the formula represented in the circuit object that describes how the characteristic circuit parameter is related to the device parameters.

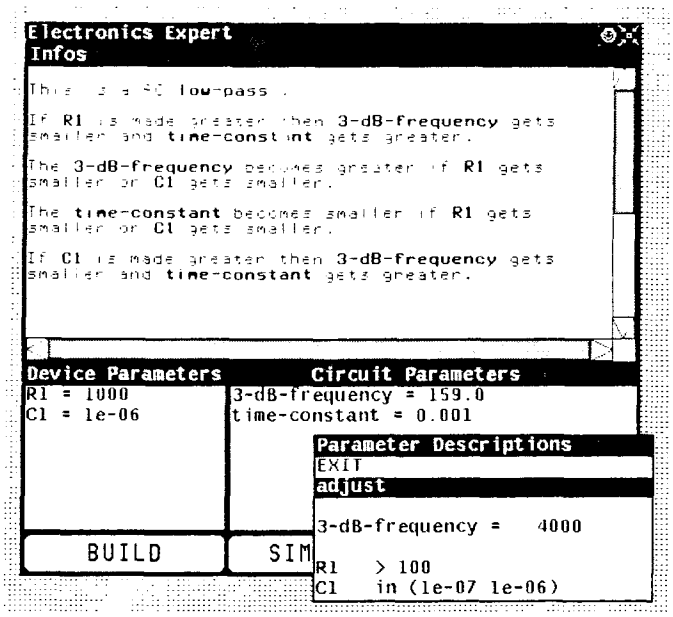

Figure 8. Adjusting the Device Parameters according to a given *3dB–Frequency*

Thus, the *Electronics Expert* is able to derive both quantitative and qualitative knowledge by combining symbolic manipulation and numerical evaluation methods.

5 Conclusions

In the domain of electronics simulation, traditional, batch-oriented systems with inadequate user interfaces are still predominant. Interactive systems using direct manipulation, however, present the problem domain in a much more natural way. We have built the prototypical system ELAB to demonstrate how to extend the desktop metaphor to a *lab metaphor* for an electronic simulation system.

The expert component ELEX has been added to the laboratory environment to support the user in performing simulated experiments and solving problems with ELAB. To cope with these tasks, both knowledge about the application domain and the particular simulated laboratory environment have been represented. This proved to be useful to make a system that is based on the visualization of a complex application domain more usable.

ELAB has been implemented in the object-oriented language ObjTalk. This turned out to be very suitable for this approach for several reasons: Object-oriented programming makes it easy to describe the communication between various instances of the problem domain in a quite natural way, and, on the other hand, supports representation of knowledge. Further, the user interface could be easily constructed using an object-oriented user interface toolkit based on the window system WLISP.

References

[Böcker, Fabian, Lemke 85]
H.-D. Böcker, F. Fabian Jr., A.C. Lemke: *"WLisp: A Window Based Programming Environment for FranzLisp"*. In *Proceedings of the First Pan Pacific Computer Conference, Volume 1*, pp 580-595. The Australian Computer Society, Melbourne, Australia, September, 1985.

[Herczeg 86]
M. Herczeg: *"Eine objektorientierte Architektur für wissensbasierte Benutzerschnittstellen"*. Dissertation, Fakultät Mathematik und Informatik der Universität Stuttgart, Dezember 1986.

[Herczeg 87]
M. Herczeg: *"Uniform Representation of Interaction Methods"*. WISDOM-Forschungsbericht FB-INF-87-30, Institut für Informatik, Universität Stuttgart, 1987.

[Herczeg 88]
J. Herczeg, M. Herczeg: *"A knowledge based electronics simulator"*. In *Applied Informatics* 30(5), pp 219-226, May, 1988.

[Hutchins et al. 86]
E.L. Hutchins, J.D. Hollan, D.A. Norman: *"Direct Manipulation Interfaces"*. In D.A. Norman, S. Draper (Editors), *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates Ltd., 1986.

[Jayaraman 84]
M. Konopasek, S. Jayaraman: *"Expert Systems for Personal Computers. The TK!Solver Approach"*. In *Byte*, pp 137-156, Mai, 1984.

[Lemke 85]
A. Lemke: *"ObjTalk84 Reference Manual"*. Technical Report CU-CS-291-85, University of Colorado, Boulder, 1985.

[Rothenberg 88]
J. Rothenberg: *"Knowledge-Based Simulation at RAND"*. In *ACM SIMULETTER* 19(2), pp 54-59, 1988.

[Shneiderman 83]
B. Shneiderman: *"Direct Manipulation: A Step Beyond Programming Languages"*. In *IEEE Computer* 16(8), pp 57-69, August, 1983.

[Sleeman, Brown 82]
D. Sleeman, J.S. Brown (Editors): *"Intelligent Tutoring Systems"*. Computers and People Series. Academic Press, London - New York, 1982.

[Smith, Weiss 88]
J.B. Smith, S.F. Weiss: *"Special Issue: Hypertext"*. In *CACM* 31(7), July, 1988.

[Vladimirescu, Newton, Pederson 80]
A. Vladimirescu, A.R. Newton, D.O. Pederson: *"SPICE Version 2G.0 User's Guide"*. Technical Report, University of California, Berkeley, 1980.