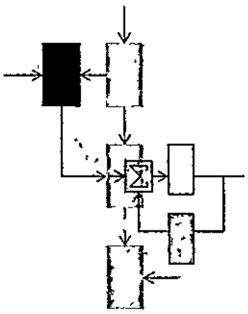
2

November, 1970

Report ESL-FR-436
M.I.T. DSR Project 71901
NASA Grant NAS 12-2208
" " NGL-22-009-019



COMPUTER AIDED ELECTRONIC CIRCUIT DESIGN
(Final Report)

Michael L. Dertouzos

Electronic Systems Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

Department of Electrical Engineering

November, 1970 Final Report ESL-FR-436

COMPUTER AIDED ELECTRONIC CIRCUIT DESIGN

(Final Report)

by

Michael L. Dertouzos

Prepared for

National Aeronautics and Space Administration
Electronics Research Center
Cambridge, Massachusetts

Electronic Systems Laboratory
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

This report describes the status of CIRCAL-2, a general purpose on-line circuit-design program. It is a final report on research sponsored by the National Aeronautics and Space Administration for a period of about five years in the area of on-line circuit design.

ACKNOWLEDGMENT

The preparation and publication of this report, including the research on which it is based, was sponsored under a series of grants to the Electronic Systems Laboratory, Massachusetts Institute of Technology. This work was begun under NASA Grant NsG-496 (Part) and continued under NASA Grants NGL-22-009-019 and NAS 12-2208. Part of this work was also supported by Project MAC, an MIT research project sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01).
This report is published for information purposes only and does not represent recommendations or conclusions of the sponsoring agencies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

A. PREAMBLE

Following the development in 1963 of a successful on-line computer utility at Project MAC, MIT, it became apparent that one of the principal applications of on-line computation was the design of electronic circuits. Accordingly, in 1965 a research program was initiated at the Electronic Systems Laboratory, MIT, under the sponsorship of the National Aeronautics and Space Administration. The central objective of this research was the effective utilization of on-line computer utilities for the design of electronic circuits.

This is a final report describing in considerable detail CIRCAL-2, a general-purpose on-line circuit-design program which is the culmination of this five-year research effort. Besides the exploration, identification and implementation of the fundamentals necessary for the development of CIRCAL-2, this five-year research has contributed several other results: Technical contributions representing a wide variety of topics in on-line circuit design have appeared in twelve reports, thirty-three technical papers and conferences and two program manuals. In addition, eighteen doctoral, master's and bachelor's theses -- the result of student participation in the project -- have been published. They contain further results of our research. The extensive conference panel participation and chairing of computer-aided circuit design sessions, by members of the project,have contributed in the dissemination of research results, and have undoubtedly influenced the development of related programs by other people. Project engineers from U.S. and overseas firms have participated in the research, as visiting staff members, and have contributed further to the dissemination of results.

On completion of this research, the CIRCAL-2 program has been converted* for system-360 use. This conversion is central to the natural consequence of our research -- the active use of CIRCAL-2 by professional circuit designers in their own environments. Already a small number of

---

* In cooperation with SofTech Inc., 391 Totten Pond Rd., Waltham, Mass. 02154

companies have committed themselves to the installation and use of CIRCAL-2 in the immediate future.

As this effort is concluded, we feel that the central objective which has dominated our research for the last five years has been successfully met. The results of our research would not have been possible without the long-range view and sustaining nature of the N.A.S.A. sponsorship. These results are now available and it is our sincere hope that N.A.S.A. will benefit directly and indirectly from their use.

The remainder of this report describes in considerable detail the program CIRCAL-2. It is part of a paper* which should appear in the Proceedings of the IEEE within the next six months.

---

* M.L. Dertouzos, G.P. Jessel, and J.R. Stinger, "CIRCAL-2; General-Purpose On-Line Circuit Design".

B. CIRCAL-2

I INTRODUCTION

CIRCAL - 2 (Circuit Analysis) is a second - generation general - purpose on-line circuit-design program. Its predecessor, CIRCAL - 1, was developed [1,2,3] in 1965, two years after the introduction of the Project MAC time-shared computer utility[4] at MIT. At least two other first - generation on-line circuit-design programs, AEDNET[5,6] and OLCA[7] were developed in about the same period, the former at MIT and the latter at Bell Telephone Laboratories.

The need for a second - generation program became apparent soon after the completion of CIRCAL - 1, from the limitations of that program, and from experience acquired in its use, and in the study of other circuit-design programs. In particular, several broad objectives were translated into the following features of CIRCAL - 2:

a. Multiple-analysis capability: The program is structured so as to permit the use of different analysis techniques. There are several reasons for this objective[8]. First it is clear that no single analysis technique can satisfy all designer requirements, such as transient, frequency or statistical analysis of circuits. Second, the amount of software "overhead" for tasks other than analysis, such as the input, modification, output and definition of information, exceeds by far the software needed for a particular analysis; consequently, it is more economical to share these overhead functions. Third, the program is flexible enough for use both by circuit designers of different specialties, and by research groups developing new analysis techniques.

   One of the consequences of this multiple analysis objective is the need for a sufficiently general common set of network elements and other basic objects which are necessary to or useful in the on-line circuit-design process. For example, means are provided for "handling" elements with hysteresis, as well as nonlinear and linear elements, and all information is structured so as to permit the choice of either time or frequency as the independent variable.

b. Effective information structures. The process of on-line circuit-design involves[8] (1) the input and modification of information which may describe, among other things, an element, a network, a source waveform or an optimization command; (2) the output of any previously inputted and possibly modified information such as a final designed network, or a response waveform; (3) the definition by a user of new objects such as element models, functions, functionals*, and optimization commands and (4) the output of informational and diagnostic comments. All of this information is structured in a sufficiently homogeneous way to permit flexibility, efficiency, and growth. For example, all information regardless of intended use, is inputted and edited by a common text editor and is stored in a common file directory on disk. In addition, the multiple analysis capability requires standardization and generality of the information needed and delivered by every analysis technique. Accordingly, the data structure that "couples" every analysis program to the system makes possible in a simple way: (1) the retrievalby analysis of all necessary information e.g. all elements connected to a particular node; (2) the return of computed results and (3) the recursive definition of new elements as networks of other elements.

Finally, commonly used objects are isolated for economy; for example, the operator which permits the definition and interpretive evaluation of functions is applicable to (1) the nonlinear characteristics of a network element e.g. a diode, or a saturating inductor; (2) the waveform of an excitation e.g. the waveform of an independent voltage source(3) the output of a function of a computed variable, e.g. the plot

* Functions with "memory" in the computing dimension which can be used to model hystersis and thermal effects.(see Section IV).

of the logarithm of a certain voltage versus frequency, and (4) the computations performed in an optimization process.

c. Efficient on-line interaction. In the course of using CIRCAL - 1 it became apparent that user-machine interaction was in some instances over-used, as for example in the need to repeat all of the parameters necessary for analysis prior to every analysis, even though only one of many parameters was changed. In other instances, interaction was under-used as for example in the inability of the user to interrupt analysis, without losing his network. In addition, there appeared to be a conflicting need to serve users ranging between two extremes - on one hand, the inexperienced user who would rather concentrate on his design, hearing from the machine about alternatives available to him at any stage of the process; and on the other hand the experienced user who does not want to waste valuable time looking at machine comments which he can predict with unerring boredom. In CIRCAL - 2 the Mode File, nested interrupt and slow/express - mode mechanisms are provided to regulate interaction.

d. Ability to define optimization procedures: In the process of design it is often necessary to adjust some parameters until a certain performance index is maximized. While several specific optimization techniques are appropriate to specific tasks, there does not seem to be one general way for optimizing circuits. For this reason, CIRCAL - 2 permits the user to define a sequence of optimization steps which normally involve successive adjustments of network parameters and analysis of the network according to his specifications. The above provision corresponds to a pseudo-user, i.e. a program that can "observe" analysis results and instigate parameter changes, performing automatically any sequence of actions that a human user could perform in order to optimize a circuit.

This paper describes main and novel features of CIRCAL - 2 and their rationale. The presentation is aimed at the communication of basic ideas rather than of programming details.

II ORGANIZATION OF CIRCAL-2

The main structure of CIRCAL-2 is shown in Fig. 1, and is explained below:

A text editor is used to input, modify and output networks, device models, functions and other blocks of information necessary for proper use of the system. The editor is not concerned with the meaning of the information that it handles but rather with its form. It is discussed in greater detail in Section III.

All information handled by the text editor is stored in a file system which resides on disk. This data is organized in six classes: Functions and Functionals (Section IV), Networks and Device Models (Section V), Mode Files (Section IX), and Defined Commands (Section X). This data is available to all operators of CIRCAL-2.

Before a network can be analyzed, CIRCAL-2 must create a data structure which represents the network and makes possible the coupling of that network with an analysis technique. Such a network data structure is normally derived from networks, device models, functions and functionals, and is described in Section VI.

In CIRCAL-2, analysis programs (ANAL i) reside on disk, and are called by the user for the analysis of a particular network. Whenever one of these programs is called, it becomes "connected" to the network data structure from where it is capable of accessing any network information necessary for that analysis; a typical request may entail all the elements connected to a given node (for setting up the conductance matrix). CIRCAL-2 analysis techniques are discussed in Section VII.

Requested analysis results are stored on disk in a standard array as they are computed. A post processor computes functions (such as the logarithm of a voltage) or functionals (such as the average of a voltage over time) of these analysis results. The post-processor output can be displayed to the user, or can be used by the defined-command operator (discussed below). The array of results and the post-processor are discussed in Section VIII.

The user, on the basis of displayed results, will normally proceed with some modifications. These may involve changing the value of network elements

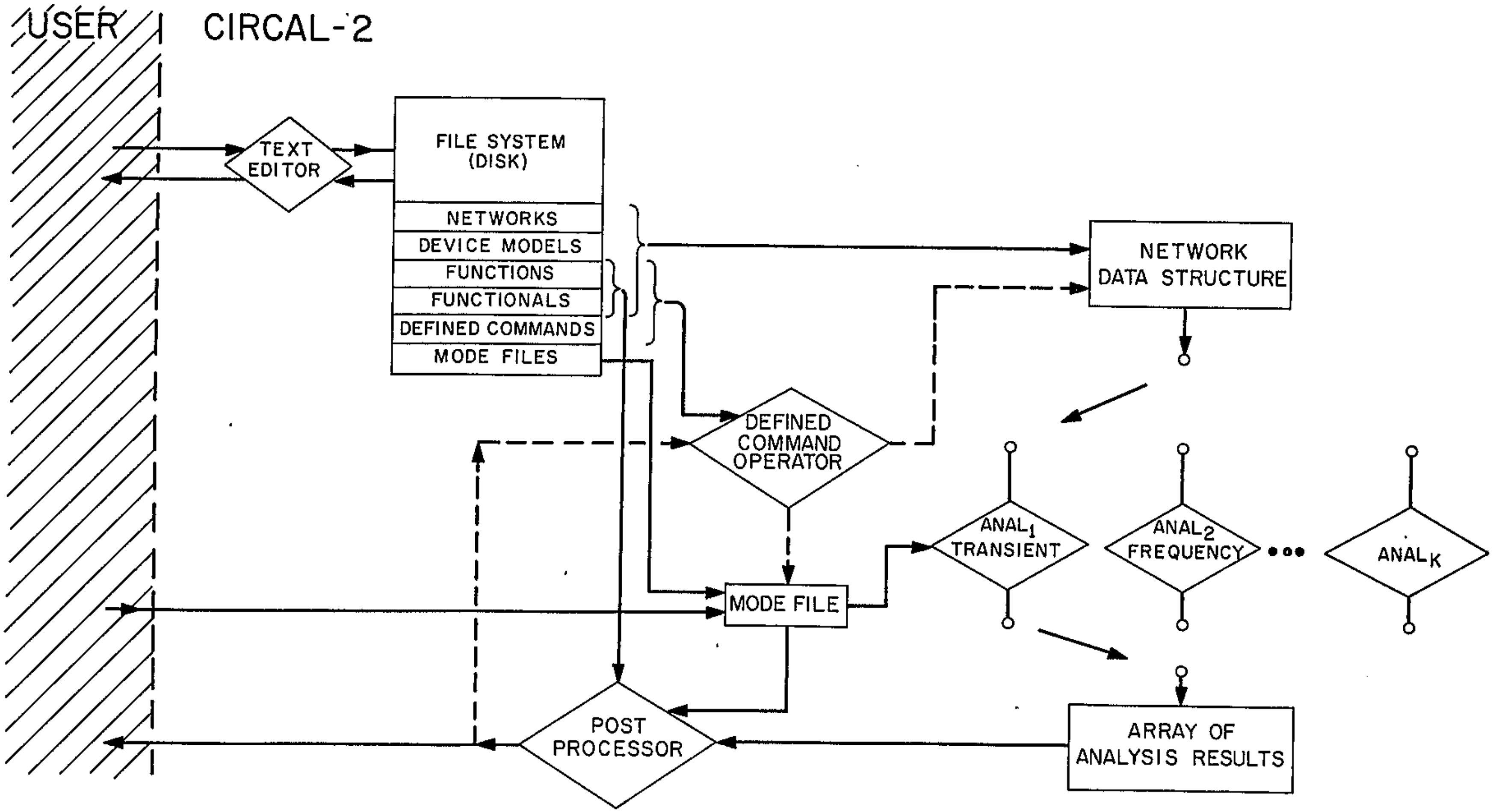


Fig. 1 Structure of CIRCAL-2

or of analysis parameters. Such incremental modifications are handled by the Mode File which stores all variables that can be changed between successive analyses of a network. The process of incremental modification and re-analysis is discussed in Section IX.

If the process of updating element values based on the results of a previous analysis is well defined, the user may create a new CIRCAL-2 command to perform the updating and re-analysis automatically without his intervention. This process is carried out by the defined-command operator on the basis of defined commands residing in the file system, and is explained in Section X.

In addition to the above, CIRCAL-2 is equipped with diagnostic error messages, computer-stored user's manual and slow/express modes of operation for inexperienced/experienced users. These additional features are discussed in Section XI.

III INPUT, EDIT AND THE FILE SYSTEM

In CIRCAL-2, as in other circuit analysis programs, there is a need for inputting and modifying descriptions of networks, network models, nonlinear elements, source waveforms, etc. In contrast to batch programs, where this information may be supplied in the form of card decks, the on-line nature of CIRCAL-2 necessitates a structure for temporary or permanent storage of these descriptions and a mechanism for inputting, editing, displaying, and deleting the stored information. Accordingly, CIRCAL-2 is equipped with a file system residing on disk, an editor and some auxiliary commands.

The CIRCAL-2 file system resembles that of an on-line operating system such as CTSS[9], but differs from the latter in that it is adapted to circuit design, and is consequently much more simple and concise. The CIRCAL-2 system consists of storage units, called files, whose size is determined solely by the amount of stored information. Two names identify each file, the first being any unique word and the second indicating the type of stored information. There are currently six types of files in CIRCAL-2, as follows:

Type	Brief Explanation
Circuit	Description of a circuit already designed or in process
Model	Description of a circuit model (e.g., Ebers-Moll model of a transistor).
Function	Definition of a function for use in element characteristics, source waveforms, etc.
Functional	Definition of a functional for description of hysteresis effects, averaging, etc.
Defined Command	Description of a command which replaces the user.
Mode File	All information which may be incrementally modified for on-line design (e.g., analysis time interval, value of a parameter, etc.)

All files are referenced through a file directory. The file manipulating commands are as follows:

Command	Action
Listf	Lists all or part of file directory as specified. (E.g., Listf* ckt displays all files of type circuit).
Printf 'file'	Prints contents of indicated file
Erase 'file'	Removes indicated file from file system.
Design 'file'	Creates or modifies indicated file, as discussed in the following paragraph.

In addition, there are provisions for locking files to prevent unauthorized use.

The creation and modification of a file is accomplished by an editor which treats all information as a string of characters without regard to content or ultimate use. All testing for syntax errors is performed by the programs which make use of the stored data. This "content insensitivity" of the editor allows system growth, since new file types can be added without requiring modification. The editor has two modes of operation, INPUT and EDIT: In the INPUT mode all characters typed are entered into the file; there is no interaction between man and machine and the user can type line after line without having to wait for a response from the system. In the EDIT mode, the first word of each line is interpreted as one of six subcommands as follows:

Subcommand	Action
Top	Top line becomes current line.
Next n	n^{th} line below current line becomes current line.
Locate 'string'	Line containing first occurrence of 'string' becomes current line.
Change 'string1'string2'	Editor replaces in current line first occurrence of 'string1' by 'string2'.
Print n	Editor prints n lines starting with current line.
Delete n	Editor deletes n lines starting with current line.

Upon completion of the input/edit process, the file subcommand is used to store the information permanently on disk. If the information consists of a function or functional description it is compiled into machine code which is also stored on disk. The compiler is explained in more detail in the following section.

IV FUNCTIONS AND FUNCTIONALS

In CIRCAL-2 functions[10] are used to describe: (a) the nonlinear characteristics of resistors, inductors and capacitors; (b) the waveforms of current and voltage sources; (c) operations on computed results (as in obtaining the logarithm of a computed voltage); (d) operations within defined commands (as in re-analyzing a network until the logarithm of a computed voltage reaches a predefined value); and (e) functionals. Functionals differ from functions in that they "have memory" along the computing dimension (time or frequency). They are used to describe (a) so-called "multivalued functions" of elements, such as hysteresis characteristics, or residual heat effects, (b) operations on computed results that require "memory", such as averaging a computed voltage over all the analysis time points and (c) operations within defined commands, such as re-analyzing a network until the time average of a computed voltage reaches a predefined value. For purposes of efficiency, functions and functionals are defined independently of their use, so that, for example, the same function can be used to describe either an element characteristic or a source waveform.

The definition of functions in CIRCAL-2 is "tailored" to the nature of source waveforms and element characteristics. Accordingly, standard functions, already defined functions, breakpoints, algebraic expressions and periodicity may be used in an arbitrarily complex way to define a new function of several variables, under the following structure:

(1) A function $y = f(x_1, \ldots, x_n)$ can be defined in "regions" formed by specifying breakpoints of the form $[x_1, y_1]$. Thus

$$E_1[1, 2]\ E_2[3, -1]\ E_3$$

defines a function. In the three regions denoted by [1, 2] and [3, -1] the function behaves as dictated by the algebraic expressions E_1, E_2, E_3.

(2) An algebraic expression is a string of constants, arguments, and functions (which are standard or already defined) connected by the usual operators +, -, * (multiply), / (divide), and ** (exponentiate). For example

[1, 4] log ((x -3) ** 2 - (x -1) ** 3) [3, -8]

means that between breakpoints [1, 4] and [3, -8], the value of the function is given by log $((x-3)^2 - (x-1)^3)$ where log is a standard function. The special case of piecewise-linear functions is handled by omitting expressions and listing only the sequence of breakpoints. Discontinuities are introduced by mismatching the value of two adjacent expressions. Omitting y_i means that $y_i = 0$.

(3) A function can also be defined by a periodic expression in the form

$$[x_1, y_1]\ [E[x_2, y_2]]\ [x_3, y_3] \text{ where } x_1 < x_2 < x_3$$

This form means that the function denoted by [0] E $[x_2, y_2]$ is shifted repeatedly starting at $x = x_1$, with period x_2 until $x = x_3$. Expressions which are periodic "forever" are handled by omitting $[x_3, y_3]$.

Functions, so defined, are given a name, which is then stated whenever the function is used.

As an example of a function description, consider the half-wave rectified sine wave of Fig. 2 which is described in CIRCAL-2 by the statement

rec(t) = [0] [sin(120*pi*t)[0.00833][0.01667]]

Here periodicity has been exploited in that only one period is described. Up to t = 0 (the first breakpoint), the value of rec(t) is zero. Between zero and 0.00833, rec(t) has the value $\sin 120\pi t$, while between 0.00833 and 0.01667, rec(t) is zero, this sequence being repeated forever.

Functionals differ from functions in that associated with each functional is a state, which may be a scalar or a vector, and which incorporates all the necessary information (along the computing dimension) about the past values of the functional and its arguments. For example, the functional description of hysteresis is shown in Fig. 3. Note that there is a single state s, calculated from earlier values of the current, which indicates the function currently characterizing the inductor.

Functionals are defined by specifying two or more functions. The first is a real-valued function of the state and other arguments and provides the

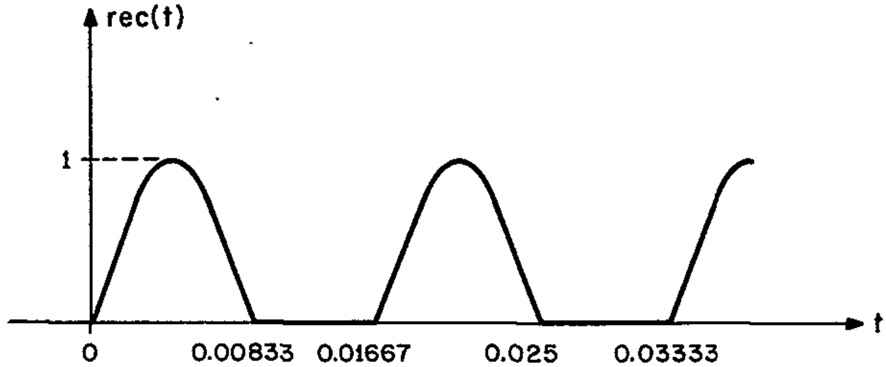
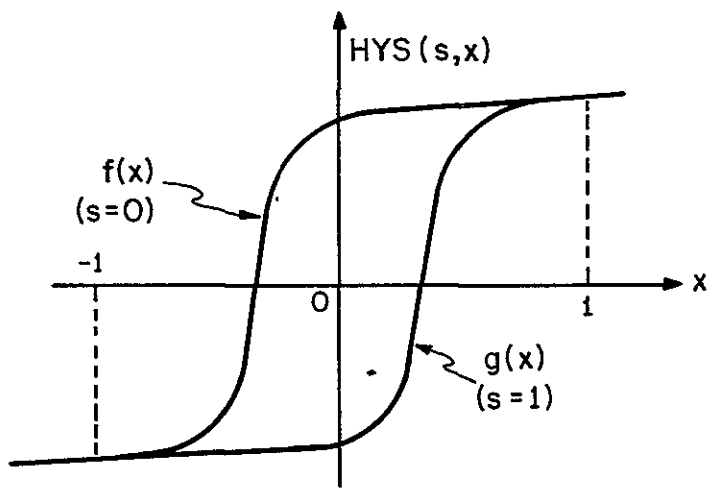


Fig. 2 Half-wave rectified sine wave



HYS(s,x) = h1(s,x), h2(x,s)

h1(s,x) = f(x) [0.5] g(x) (output function)

h2(x,s) = 1[-1] s [1] 0 (state-update function)

Fig. 3 Hysteresis functional and its CIRCAL-2 Description value of the functional. The remaining functions update each component of the state vector as analysis progresses. All of these functions are of the type described above and permit any number of arguments. The initial state of a functional is specified for each instance of that functional, thereby permitting a single block of compiled machine code to be used for several elements.

All user-defined functions and functionals in CIRCAL-2 are processed by a compiler which is automatically entered via the text editor when a function or functional description is filed. (This is the only case where information entered through the text editor is interpreted immediately). From the function or functional description, the compiler generates relocatable machine code which is stored on disk in a file separate from the CIRCAL-2 file system and thereby inaccessible to the user. If errors are found in the description, standard CIRCAL-2 error diagnostics are issued. Whether the compilation is successful or unsuccessful, the program returns to command status.

Written expressly for CIRCAL-2, the compiler is tailored to the needs of the CIRCAL-2 program and produces machine code which is comparable in efficiency to that produced by a high-level language compiler (the half-wave rectified sine wave discussed earlier compiles to seventy-two machine instructions on the IBM 7094). Since over a given analysis, functions and functionals may be called several hundred times, generating and executing compiled machine code is more efficient than interpretively executing the function or functional description, by as much as a factor of ten.

V NETWORK ELEMENTS

In developing a computer-aided design system, the choice of a set of network elements has, in the past, been dictated by the prescribed analysis routine. On the other hand, the multi-analysis structure of CIRCAL-2 means that even if a set of elements compatible with all present analysis routines were specified, no guarantee would exist that this set would work with a future analysis. For this reason the set of network elements chosen for CIRCAL-2 had to be general enough to allow modeling of almost any electrical network to almost any desired detail.

In CIRCAL-2, analysis is concerned with the relationship between the network variables

charge, voltage, current and flux

and their variation with the computing dimension, e.g., time, frequency or spatial distribution. An element is defined by an explicit function or functional which relates pairs of branch variables — a resistor relates current and voltage, a capacitor relates charge and voltage and an inductor relates flux and current. In addition to one of the branch variables, the domain of this function or functional may include:

(1) network variables of other branches

(2) the computing dimension

(3) any parameters, and

(4) functional "state" variables

Inclusion of only the branch variable makes possible the description of linear and non-linear RLC's and constant sources. The addition of other network variables makes possible the description of controlled sources (linear and non-linear) and parasitic coupling effects (e.g., in integrated circuits).

Inclusion of the computing dimension gives rise to time-varying, frequency-varying, or spatially-varying elements and sources. The computing dimension is distinguished from the set of parameters since it varies during analysis. A variable such as frequency may be considered as the computing dimension in one analysis (frequency) and as a parameter in another (transient).

Parameters can be used to denote tolerances, aging factors, the gain of a transistor, the value of an element, i.e., any variable that remains constant throughout a single analysis "run". The use of parameters is crucial to the incremental modification process inherent to on-line design, since the user (or the defined-command operator) normally varies parameter values between successive analyses.

The inclusion of the above three types of arguments in the domain of an element description is sufficient for most existing analysis routines. However, elements whose behavior depends on the values of the branch variables over some previous interval of the computing dimension cannot be modeled. The most common such effects are hysteresis and heating. It is for the description of such elements that CIRCAL-2 allows the use of functionals. The functional "state" variables of (4) above, are precisely those variables that retain memory along the computing dimension as discussed in Section IV. For the inductor with hysteresis of Figure 4, the state is given by the value of the polarization, and is modeled using the functional of Fig. 3.

The format which makes possible the description of all elements and sources is as follows:

n_1 n_2 element-name   value

where n_1, n_2 are node numbers corresponding to element/source terminals, and element-name and value are as indicated in Table 1. Observe in Table 1, that the frequently-occurring linear RLC's and sources have a simple representation, e.g.,

1   2   R19   220

means a linear 220Ω resistor connected between nodes 1 and 2. Observe further that non-linear elements conforming to the above discussion can also be defined by this format. For example

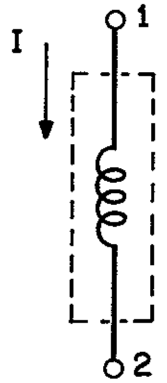
1   2   NR19   F(V(NR19), V(NR14), t, tol)

means a non-linear resistor (NR19) connected between nodes 1 and 2 whose current* is given by function F (defined elsewhere), whose arguments are the branch voltage of NR19, the branch voltage of another element NR14, the computing dimension (time t),

* The program automatically detects that this is a voltage-controlled resistor since its own branch voltage is the first argument of F.

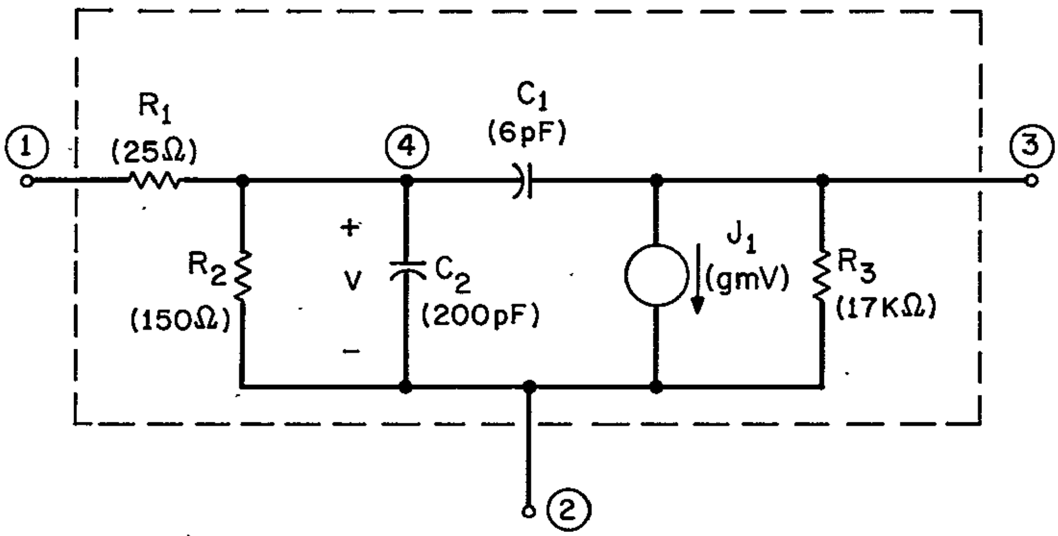
Table 1 CIRCAL-2 elements

	ELEMENT NAME	VALUE
LINEAR	R - resistor L - inductor C - capacitor S - switch J - current source E - voltage source	Element or source value is one of the following: 1 - numerical constant 2 - parameter 3 - function whose domain may include: a - numerical constant b - parameter c - computing dimension *For controlled sources the value must be preceded by the controlling variable and associated branch element.*
NONLINEAR	NR-resistor NL-inductor NC-capacitor NS-switch NJ-controlled current source NE-controlled voltage source	Element characteristic described by function or functional whose domain must contain a - one local branch variable and may include b - parameters c - computing dimension d - other element variables e - functional state variables *For controlled sources, the domain must <u>not</u> include a local branch variable.*




```
1   2   NL1  HYS(1,I)
```

Fig. 4 Nonlinear inductor and its CIRCAL-2 description




```
XNODES  3  1  2  3
ARGS   1   GM
 1     4   R1   25.
 4     2   R2   150.
 4     2   C2   2. E-10
 4     3   C1   6. E-12
 3     2   R3   17. E3
 3     2   J1   V(C2)  GM
```

Fig. 5 Hybrid-pi model of a silicon transistor with its CIRCAL-2 description and a parameter tol.

CIRCAL-2 also allows the user to define his own models or nested structures using any of the above standard or defined two-terminal elements, and recursively, other nested structures. The format which makes this possible is as follows:

$$n_1 \;\; n_2 \;\; \ldots \;\; n_k \qquad \text{model-name } (arg_1, arg_2, \ldots, arg_m)$$

Here $n_1 \ldots n_k$ is an ordered list of nodes to which the model is connected. Models are defined in the same way as networks, except that external nodes and arguments must be specified. These arguments can only be parameters of elements making up the model. Observe that models may be used in the construction of other models and that once a network is defined to be a model, all of its internal network variables become inaccessible. This is as it should be, since network elements outside the model should not depend on variables internal to the model. A transistor model and its CIRCAL-2 description is shown in Fig. 5.

VI THE NETWORK DATA STRUCTURE

The network data structure is essentially an ordered representation of all the information necessary for analysis of that network. The data structure facilitates access of such information as "all elements connected to a given node", or "all nodes of a given type". Analysis of the circuit is thereby simplified by eliminating some of the time-consuming searches and tedious bookkeeping tasks from the analysis routine.

The multiple-analysis capability of CIRCAL-2 requires a general network data structure that places no constraint on any one of the several possible analysis routines. To establish the features which should be included in such a general data structure, the following circuit-analysis programs were examined: ECAP[12], NET-1[13], CIRCUS[14], SCEPTRE[15], MISSAP[16], OLCA[7], AEDNET[6], and CIRCAL-1[1]. Table 2 summarizes the data structure features of these programs in CIRCAL-2 terminology (explained later in this section) by "1" entries at the appropriate intersections. "2" entries indicate those features which, if present, could make analysis more efficient. "3" entries indicate common features which would permit implementation of any one of the above analyses. These "3" entries are the features included in the CIRCAL-2 data structure.

The CIRCAL-2 data structure, unlike the fixed-length arrays used by the first five programs of Table 2, makes use of variable-length blocks of storage which are dynamically allocated according to need. The data structure consists of <u>beads</u> and <u>pointers</u>. Beads are blocks of contiguous memory registers which contain information pertaining to a network, an element, a node, etc. Pointers are addresses of beads and serve to interconnect beads to form <u>strings</u> (Fig. 6). Each string corresponds to a feature derived from the "3" entries in Table 2. For example the string of all resistors interconnects, with pointers, only those element beads which correspond to resistors. Beads will generally be members of more than one string, e.g., a bead describing a resistor will be included both in the resistor string and in the string of all elements. To save storage, the CIRCAL-2 strings are grouped into four classes such that no two strings of a class will have a bead in common. The CIRCAL-2 strings and classes are given below:

<u>Class I Strings</u>

Elements

Internal nodes

External nodes

Table 2 Data structure features of several CAD programs

Data Structure Features \ Programs	ECAP	NET-1	CIRCUS	SCEPTRE	MISSAP	OLCA	AEDNET	CIRCAL-1	CIRCAL-2
String of all elements						1			3
String of all nodes								1	
String of all resistors	1	1	1	1	1	2	1	2	3
String of all inductors	1	1	1	1	1	2	1	2	3
String of all capacitors	1	1	1	1	1	2	1	2	3
String of all transformers									3
String of all models		1	1	1	1		1	2	3
String of all elements described by function				1	1		1	2	3
String of all elements described by functional					1				3
String of all linear elements					2				
String of all piecewise-linear elements	2								
String of all non-linear elements					2				
String of all fixed-by-user elements [a]				2	2	2			
String of all variable-by-user elements [b]				2	2	2			
String of all variable-by-program elements									
String of all independent voltage sources	1	1	1	1	1	2	1		3
String of all independent current sources	1	1	1	1	1	2	1		3
String of all dependent voltage sources				1	1	2	1		3
String of all dependent current sources	1			1	1	2	1		3
String of all external nodes		1	1	1	1		1		3
String of all internal nodes		1	1	1	1		1	1	3
String of all linear nodes [c]								2	
String of all non-linear nodes [d]								2	
Function/functional use	2	1	1	1	1		1	1	3
Pointers from element beads to node beads	1	1	1	1	1	1	1		3
Pointers from element bead to external nodes of model								2	
String of all elements connected to a node								1	3
Present node voltage								1	
Present branch charge or voltage	1	1	1	1	1	1	1		
Present branch flux or current	1	1	1	1	1	1	1		
Element type						1		1	3
Polarity of element								1	3
String of all time or frequency varying elements	2	2	2	2	2		2	2	3
Number of members of each string	2	2	2	2	2	2	2	2	3

Footnotes to Table 2

a. Value of element is fixed

b. Value of element specified by parameter

c. Nodes with only linear elements connected to them

d. Nodes with one or more non-linear elements connected to them

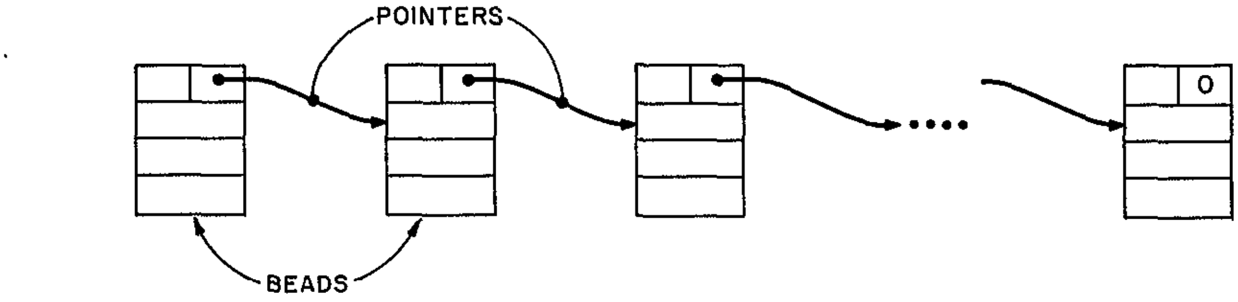


Fig. 6 A typical string of the data structure

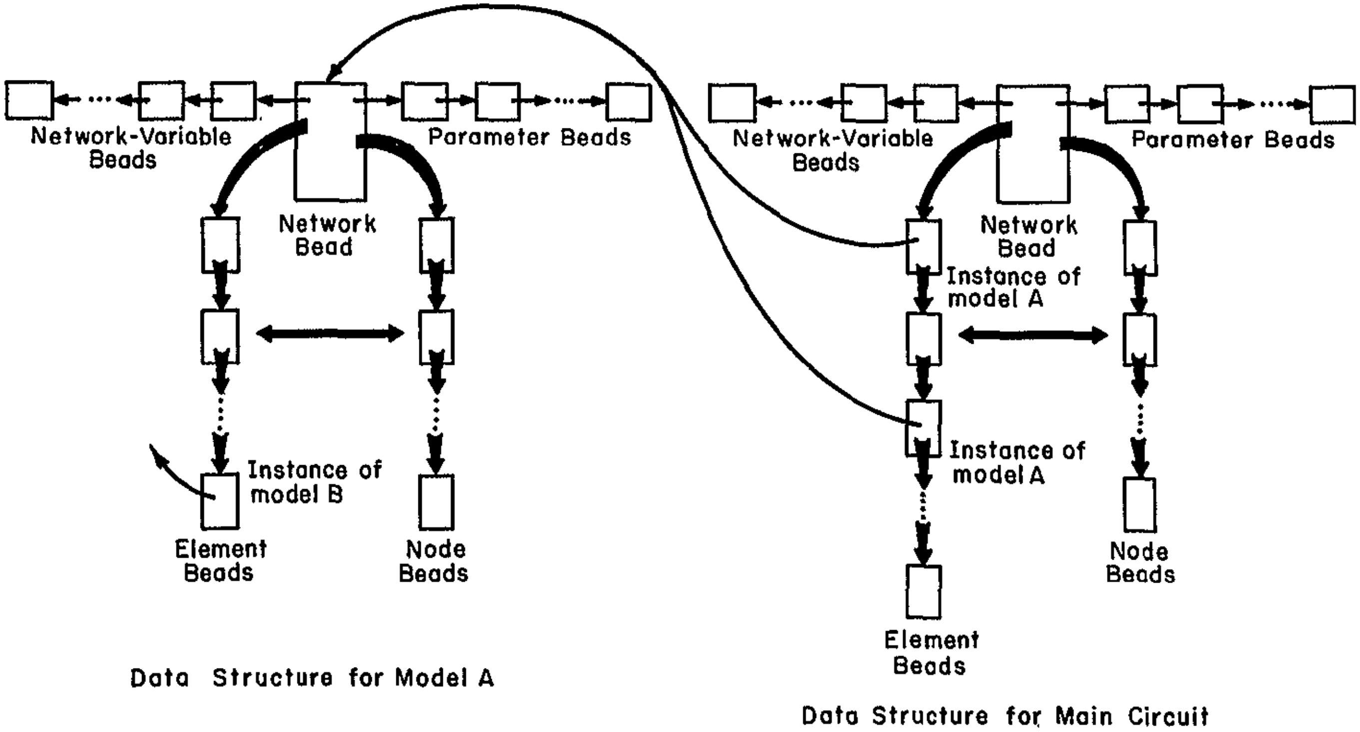


Fig. 7 The interconnection of the five bead types to form the data structure (heavy arrows represent more than one pointer)

Class II Strings

Resistors

Inductors

Capacitors

Transformers

Independent voltage sources

Independent current sources

Independent switches

Dependent voltage sources

Dependent current sources

Dependent switches

Models

Class III Strings

Elements dependent on computing dimension

Class IV Strings

Elements described by functions

Elements described by functionals

To create a network data structure from the network file, or to access information from the data structure for analysis, certain operations must be performed on these strings. There are four such basic operations, which, when appropriately combined, allow a bead to be inserted in, or removed from a string; a bead to be located; and a pointer to the next bead to be obtained.

The form of the CIRCAL-2 data structure is shown in Fig. 7. Note that this structure is "recursive" in that the data structure for a model is of exactly the same form as the data structure for the main circuit. Even though there may be many instances of a model, typically characterized by different argument values, there is only one data structure for the model, pointed to by each such instance. No limit* is placed on the number of levels to which models may be nested.

There are five types of beads in the CIRCAL-2 data structure: (1) network beads; (2) parameter beads; (3) network-variable beads; (4) element beads; and

---

* Other than core-memory size

(5) node beads. A description of each is given below.

The main circuit and each model are characterized by a network bead which serves as the origin of every string in that network. For each such string, the network bead contains the number of beads in that string and a pointer to the first bead on the string. The form of the network bead is shown in Fig. 8.

For each distinct main-network parameter or model argument there is one parameter bead (Fig. 9) containing the name of the variable and space for the value of that variable.

Elements described by a function or functional generally depend on one or more network variables (voltage, current, flux, charge). For each such network variable there is a network-variable bead (Fig. 10) which contains the names of the network variable and the element with which it is associated, as well as space for the value of that variable. Since this value will change during analysis, it must be updated by the analysis routine at each step.

To each element or model instance in the network there corresponds an element bead (Fig. 11) containing information pertinent to that object, e.g., argument values for a model instance. The element bead varies in length, depending on such things as how many terminals the element has or whether it is described by a function. Note that the bead contains pointers to several next-in-the-string elements, since an element bead will generally lie within several strings. When the value of an element is specified either by a constant or by a parameter, the pointer to the element value points in the first case to a word containing the numerical constant, and in the second case to the bead on the parameter string which contains the value of the parameter. If the element is described by a function or functional, each argument pointer points to a word containing the value of that argument, which in turn may be either a constant, a parameter, or a network variable. If the element is a model instance, then the argument values of that instance must be copied by the analysis routine into the common data structure for that model so that the model may be analyzed.

Finally, for each node in the circuit, there is a node bead (Fig. 12) containing information about that node, and a pointer to the next bead on the node

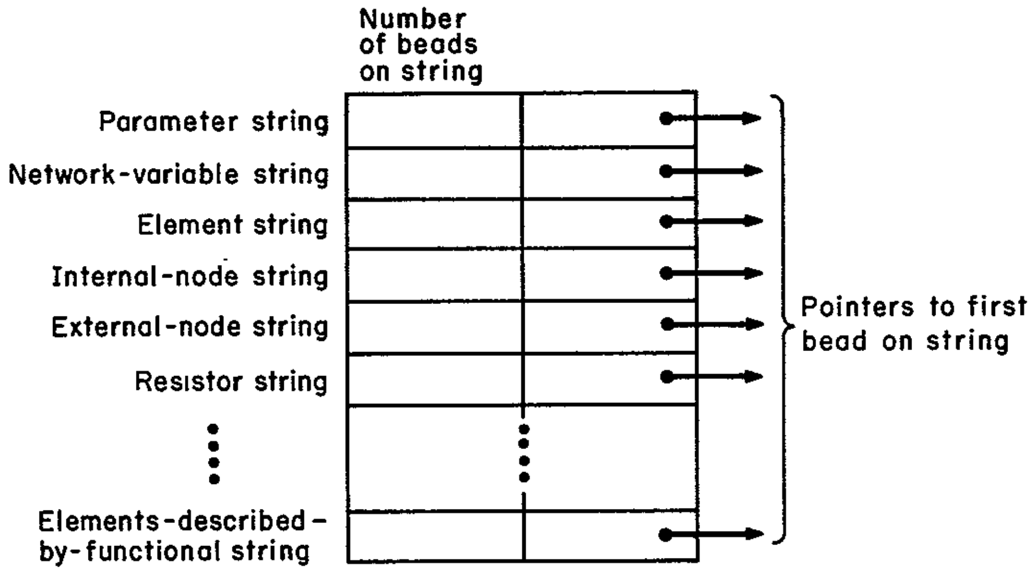


Fig. 8 The network bead

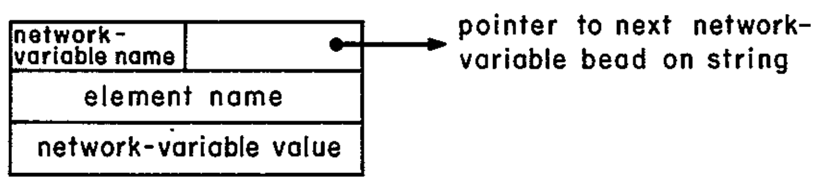


Fig. 9 The parameter bead

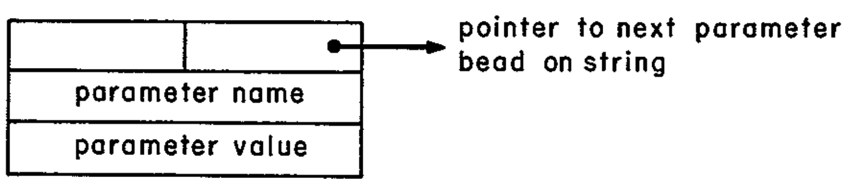


Fig. 10 The network-variable bead

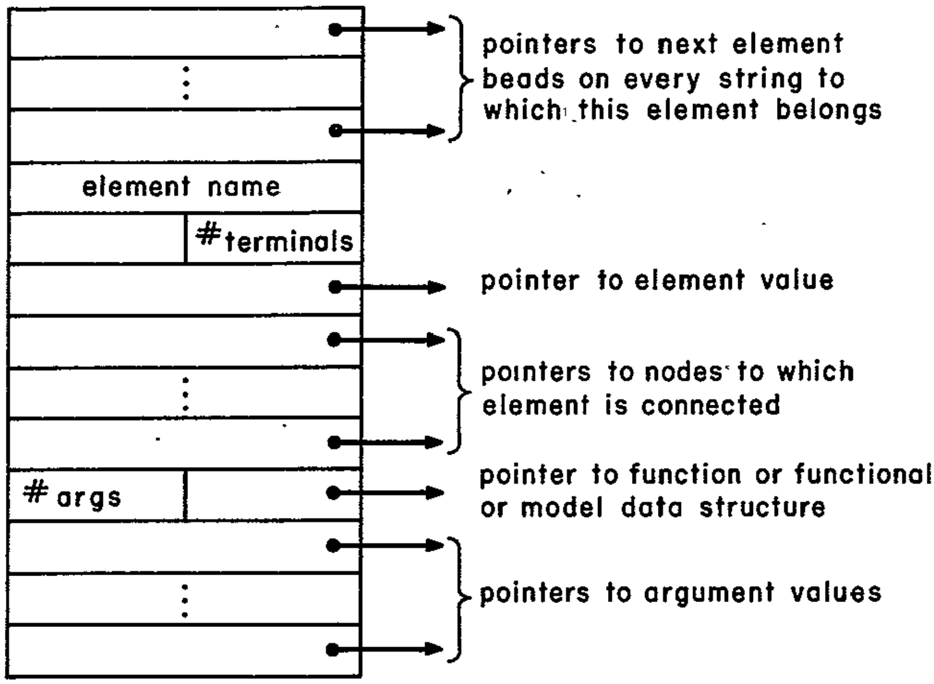


Fig. 11 The element bead

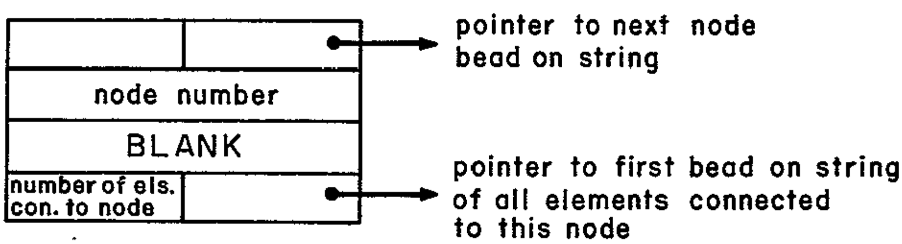


Fig. 12 The node bead

string. The BLANK word shown in Fig. 12 is for use by most analysis routines, which find it convenient to re-number the nodes. The last word in the node bead is the origin of the string of all elements connected to that node--useful in nodal analysis.

VII ANALYSIS

Analysis can be viewed as a function, which given an electrical network returns an array of numbers. Typically, such an analysis will consist of two closely-coupled operators. The first of these operators performs a topological examination of the network to formulate a set of equations, e.g., a conductance matrix, a cutset matrix or state equations—a primary purpose of the CIRCAL-2 data structure, described in the previous section, is to facilitate this formulation task. The second operator involves solution of the formulated equations and usually draws upon standard mathematical techniques such as matrix inversion, iteration and integration. This generality permits the implementation within CIRCAL-2 of less conventional analyses, such as the computation of the sum of all resistors in a network (useful in calculating the area of integrated circuits).

Recall that the multi-analysis structure of CIRCAL-2 requires maximum isolation of the analysis routines from the rest of the system, even at the expense of some efficiency. Accordingly, each analysis program couples to the rest of CIRCAL-2 through: (1) a network data structure whose formation is independent of analysis; (2) the Mode File (discussed in Section IX) which handles all user-communicated analysis data in a uniform manner; and (3) an output data structure for the identical treatment of computed analysis results.

New analysis programs may be incorporated without any changes to the system unless it is very desirable to have some new property of the network represented as a data structure string, in which case minor modifications to the data structure are necessary (see Section VI). Note that in general, such modifications will not be required, since the data structure in its most primitive form is one-to-one with the network, and particular properties of an element or node can be found by searching the string of all elements or nodes.

inclusion of a statistical and d.c. analysis as well as several special-purpose routines[19,20] developed in the course of our research. The addition of new analysis programs is relatively easy, and it is envisioned that users of the program will augment it with special-purpose analysis techniques particular to their needs.

The following discussion refers to a straightforward frequency analysis technique written for CIRCAL-2. It illustrates some of the features common to all analyses, and in particular emphasizes the interface between analysis and the data structures.

The recursive nature of the network data structure is exploited through use of recursive procedures* which treat each model in a manner identical to that of the main network. Any discussion with reference to the main network is therefore applicable to each instance of a model. Initially, the network bead of the main circuit (and of each model) is read to: (1) check for disallowed elements and (2) determine the dimension of the network. If a disallowed element is found, analysis is not continued. The dimension of the network, determined by counting the nodes of the main circuit and all internal nodes of models, is used to dynamically allocate core memory for the admittance matrices. This means that no artificial constraint (i.e., fixed arrays) is imposed by the system on the network size. Reading of the data structure is performed by a single procedure which, given as arguments the name of a particular program (created for that analysis) and starting location of a string, applies this program to each member bead of the string. The strings of resistors, capacitors, inductors and sources are scanned in this manner to generate the nodal conductance, capacitance and reluctance matrices. At each frequency, specified by the user in the Mode File, these matrices are combined to form a complex admittance matrix. The desired node voltages are then calculated by Gaussian reduction and stored on disk using an output routine common to all analyses and discussed in the following section.

Approximately three man-weeks were required to implement this analysis. This is much less than the time needed to develop a completely new circuit-design program to do the identical frequency analysis.

At present, CIRCAL-2 contains a general-purpose nonlinear transient and two frequency analysis routines, one of which uses standard Gaussian reduction and one which employs sparse matrix methods. Current plans also call for

---

* Part of the AED[17,18] programming system.

VIII THE OUTPUT OF ANALYSIS RESULTS

The availability of multiple analysis routines in CIRCAL-2 makes it necessary for the analysis results to be in a standard form, enabling a single output processor to operate on the results of any analysis program. These results are stored in two arrays, and are processed by two operators.

As analysis proceeds, the computed results are inserted into the output array (Fig. 13) in blocks, each block B_j corresponding to an analysis point in time t_j, or frequency f_j*. The size of these blocks is determined by the variables s_1, s_2, ... that the user has requested to be saved (through the Mode File). For example, the user may have specified that the voltage, V_{19}, from node 19 to ground and the current, $I(R_{11})$, through resistor R_{11} be saved. In this case, s_1 would be V_{19} and s_2 would be $I(R_{11})$, both stored versus time if a transient analysis were performed.

Once analysis has been completed, the data stored in the output array is processed by a CIRCAL-2 function/functional operator which computes standard or user-defined functions and/or functionals of the analysis data, as specified by the user through the Mode File, and stores the results in the plot/print array. For example, the user may wish to see the logarithm of V_{19} versus $I(R_{11})$, in which case r_1 of Fig. 13 would be log (V_{19}) and d_j would be $I(R_{11})$, both evaluated at t_j, and stored for all computed analysis points.

The data stored in the plot/print array is then passed through the plot/print operator which presents the results of analysis to the user in the form of a graph or a table of values.

Each variable, s_i, to be saved is specified by at least two and at most three terms A, B, C. A may be any name which analysis can recognize; B may be a name or an integer; and C must be an integer. For example, V_{19} would be specified as V 19, while $I(R_{11})$ would be specified as I R11. The specification I TRANS6 9 means the current entering the 6^{th} instance of model TRANS at network node 9. Clearly, this method of specification is quite general,

* Or any other computing dimension such as node number or space.

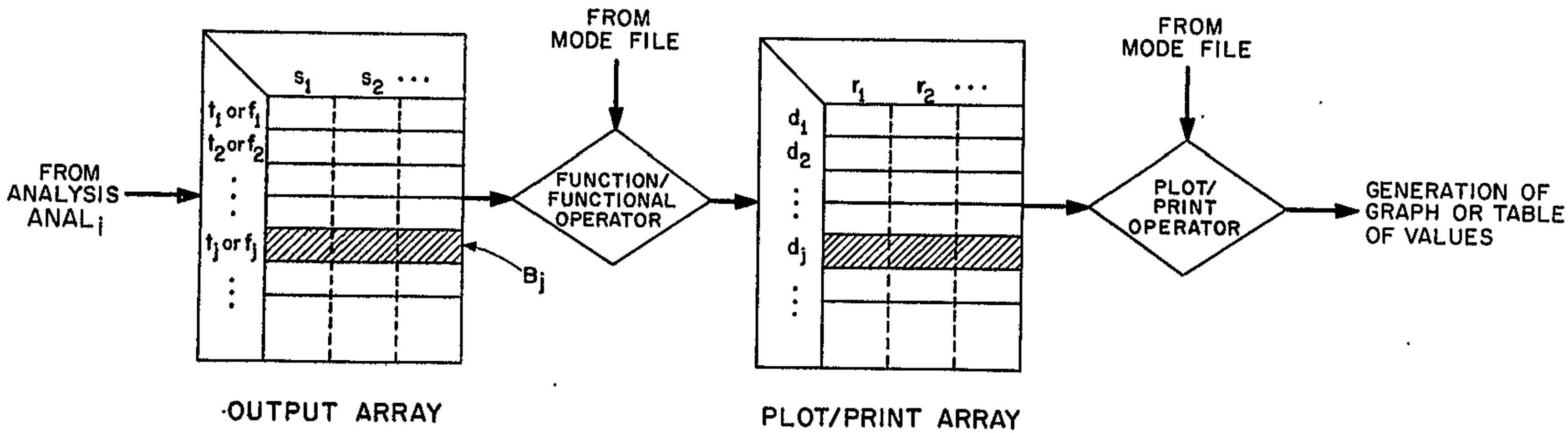


Fig. 13 Output data structure and processors

and is restricted only by those objects which the analysis program can recognize.

Any CIRCAL-2 functions or functionals $f_1, \ldots, f_k$ of the saved variables and the computing dimension may be calculated and stored as columns $r_1, \ldots, r_k$ in the plot/print array of Fig. 13. The rows are ordered along the computing dimension, and the values $d_1, \ldots, d_m$ are computed by a function or functional g whose arguments may be the same as those of f_i. Thus, for example, the plot/print array may store a voltage versus another voltage, a power versus a voltage, a time integral of a voltage versus time, and so forth. When a function is not explicitly specified, the identity function of the quantities to be plotted or printed is automatically invoked.

Both the output array and the plot/print array are stored on disk, rather than in core. Thus, the amount of data saved for output is independent of the amount of dynamically allocatable core storage available, permitting the analysis of a larger circuit. This method of storing data on disk has its disadvantages, however. Ordering the data for plotting purposes is difficult unless such data is brought into core. Moreover, additional time is required to transfer the data between core and disk, and to conduct the associated disk operations. This method is practical when the computed analysis results occupy more than the amount of core required to buffer the disk.

An example of the use of functions in the output of analysis results is shown in Fig. 14, where the analysis results are printed as a table of values and the magnitude of these results is plotted using the standard CIRCAL-2 magnitude function. Comments appear on the right in italics.

```
COM- repeat print  2  vi 1  vr 1

VALUES FOR
  1 VI 1
  2 VR 1
VS F
```

The REPEAT command (described in Section IX) is used to obt a printout of the real and im inary parts of the voltage at node 1 with respect to ground The independent variable (IV) is frequency, while 1 and 2 a the imaginary and real parts the voltage respectively.

IV	1	2
1.999E-01	-1.465E+00	2.532E-02
3.999E-01	-5.443E-01	6.332E-03
5.999E-01	-1.535E-01	2.814E-03
7.999E-01	1.047E-01	1.583E-03
9.999E-01	3.100E-01	1.013E-03
1.199E+00	4.887E-01	7.036E-04
1.399E+00	6.522E-01	5.169E-04
1.599E+00	8.063E-01	3.957E-04
1.799E+00	9.541E-01	3.127E-04
1.999E+00	1.097E+00	2.533E-04
2.199E+00	1.237E+00	2.093E-04
2.399E+00	1.375E+00	1.759E-04

```
COM- repeat plot 1 mag(vi 1 vr 1)  pi = .1

GRAPH OF
  1 MAG(VI 1,VR 1)
VS F
VAR  SCALE
     FACTOR
```

The REPEAT command is used tc obtain a plot of the magnituc (horizontal axis) of the abov analysis data versus frequenc (vertical axis), using the standard CIRCAL-2 magnitude function. The plot increment (pi) is changed to .1 for thi plot.

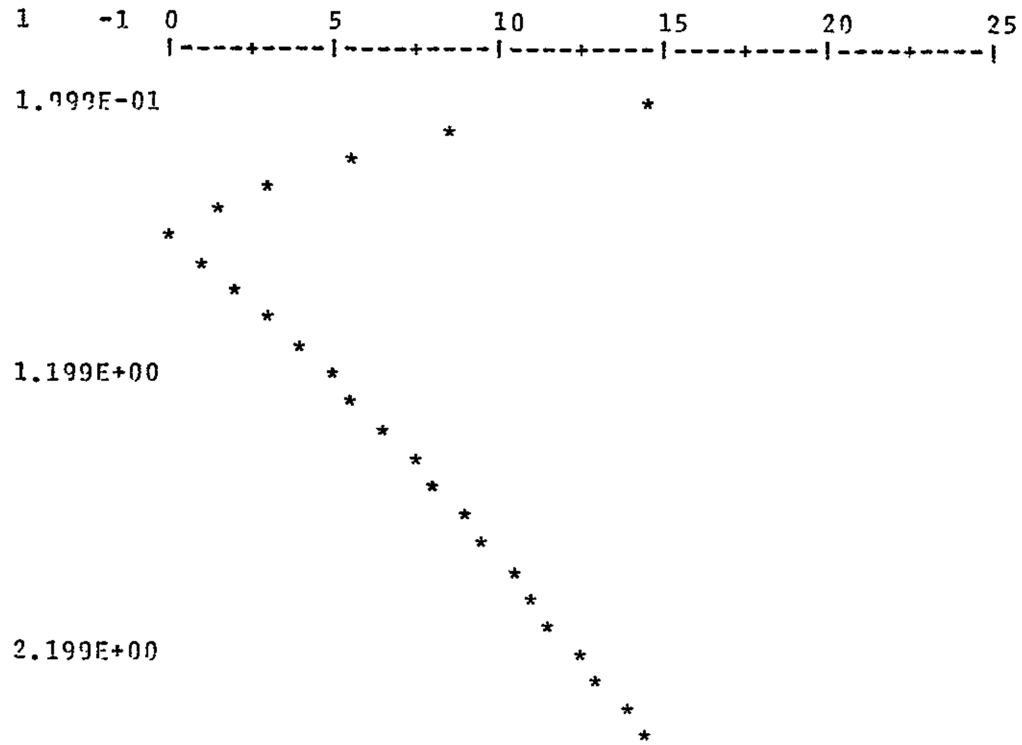


Fig. 14 Use of a function in the output of analysis results

IX INCREMENTAL MODIFICATIONS AND RE-ANALYSIS; THE MODE FILE

Design in CIRCAL-2 is conducted through two possible avenues. One involves the specification of an optimization program which automatically adjusts parameters until a desired performance is achieved — this is discussed in the following section. The other and, by far, more common approach involves the on-line interaction of program and user, with the former conducting analysis and the latter evaluating analysis results and making appropriate modifications. Experience with on-line circuit design programs[2,5,6,7,8] has shown that the circuit designer spends a great deal of his time requesting re-analysis of the circuit, each time having to specify a long list of analysis parameters. In addition, a separate command or commands must be used merely to change the value of an element. These actions not only increase substantially the amount of interaction between the program and the user, but they also require the user to devote much of his thinking to the choice and use of the proper command sequence and command arguments. In CIRCAL-2, these limitations have been greatly reduced through use of the Mode File. This file, which contains all auxiliary information necessary for analysis and output is created prior to the first analysis of the circuit. Thereafter, the user need only specify the information he wishes to change for each re-analysis.

The Mode File contains values for the analysis and output parameters. In particular, these quantities are:

(1) network parameters (values of linear elements, arguments of models, functions and functionals — all of which were previously specified by variables rather than by numerical constants)

(2) the name of the analysis routine to be used in analyzing the circuit

(3) the analysis start point

(4) the analysis end point

(5) the analysis increment

(6) the computing dimension (e.g., time or frequency)

(7) a list of variables whose values are to be saved for output

(8) the output start point

(9) the output end point

(10) the output increment

(11) the type of output (plot or print)

(12) a list of functions and/or functionals of the saved variables whose values are to be either plotted or printed, and

(13) a list of the analysis routines which the user has indicated he will use on the circuit.

Observe that the plot/print parameters need not be identical to the analysis parameters, e.g., only every n^{th} computed point may be plotted, and the beginning and end of a plot may be different than the beginning and end of analysis.

Once the Mode File has been specified, the user indicates his modifications by typing

REPEAT $P_1 = v_1, P_2 = v_2, \ldots$

where P_i is the name of a variable from the above list, and v_i is its new value. CIRCAL-2 then checks to determine if re-analysis is necessary and proceeds to fullfil the request. In particular, changes in the processing of computed results are performed without re-analysis.

All network modifications conducted in this way must be done by network parameter changes. Thus to change the value of an element, the nature of a waveform, or a nonlinear characteristic, the user need only type the command REPEAT followed by the new value of the corresponding network parameter. This of course presumes that the user had the foresight to assign variables to his desired adjustments. If he did not, then the DESIGN command must be used to alter the network description to include these variables. Once the network parameters have been updated, analysis of the circuit is repeated and analysis results are processed according to the output specifications in the Mode File. The user may then make further modifications through the REPEAT command, continuing in this manner until satisfactory results are obtained. An example of the modifications possible through use of the REPEAT command is shown in Fig. 15, with appropriate comments in italics.

To help the initialization of the Mode File, CIRCAL-2 is equipped with a permanent Mode File containing preset values for most of the Mode File parameters (a few of the parameters cannot be preset, such as the variables to be

```
COM- repeat  k = .5  ps = 1. pi = .2
```

The REPEAT command is used to change the values of the network parameter k, the output start point ps, and the output increment pi. All other mode file parameters retain their previously assigned values.

```
VALUES FOR
  1 VI 1
  2 VR 1
VS F

    IV              1             2
 9.999E-01      3.100E-01     1.013E-03
 1.199E+00      4.887E-01     7.036E-04
 1.399E+00      6.522E-01     5.169E-04
 1.599E+00      8.063E-01     3.957E-04
 1.799E+00      9.541E-01     3.127E-04
 1.999E+00      1.097E+00     2.533E-04
 2.199E+00      1.237E+00     2.093E-04
 2.399E+00      1.375E+00     1.759E-04
```

```
COM- repeat  k = .2
```

Repeat, changing only the value of k.

```
VALUES FOR
  1 VI 1
  2 VR 1
VS F

    IV              1             2
 9.999E-01     -1.674E-01     6.332E-03
 1.199E+00      9.086E-02     4.397E-03
 1.399E+00      3.112E-01     3.230E-03
 1.599E+00      5.079E-01     2.473E-03
 1.799E+00      6.988E-01     1.954E-03
 1.999E+00      8.587E-01     1.583E-03
 2.199E+00      1.020E+00     1.308E-03
 2.399E+00      1.176E+00     1.099E-03
```

Fig. 15 Use of the REPEAT command

saved for output, and <u>must</u> be specified by the user prior to the first analysis). All other parameters need only be specified if a value other than the preset value is desired.

CIRCAL-2 automatically assigns a distinct Mode File to each circuit that is analyzed. Each such Mode File is saved so that it does not have to be re-created for future analyses of the same circuit during a CIRCAL-2 session, even though other circuits may have been analyzed in the meantime. Mode Files may also be independently created through the DESIGN command and stored on disk.

There are two modes in which updating of the Mode File may be done — <u>slow</u> and <u>express</u> — determined by the user. The slow mode is intended for the inexperienced CIRCAL-2 user. In this mode, the user is asked successively to specify the value of each item in the Mode File. The user responds by either typing a value or by carriage return if he does not wish to change the value. The express mode is intended for the more experienced user who is familiar with the contents of the Mode File and the manner in which values are specified. In this mode, the user is allowed to type all values at one time, thereby substantially reducing the interaction time between program and user.

The Mode File, after it has been updated, is examined by the program to insure that values have been specified for all variables that could not be preset. Values for any such variables that have not been specified are then requested by the program.

X AUTOMATIC MODIFICATION AND OPTIMIZATION; DEFCOM

The defined-command ability (Defcom) of CIRCAL-2 makes possible the definition by the user of a program which automatically controls the analysis and modification of a circuit until certain objectives are met. In other words, when the user understands precisely the steps involved in desiging a class of circuits, he may create a "Defcom" which will automatically execute these steps. Thus, a Defcom acts as a pseudo-user, in that it "observes" computed results and makes network modifications in accordance with the algorithm implemented by that Defcom.

The purpose of this defined-command ability is to make possible the specification of any algorithmic optimization procedure. The phiosophy behind it is to provide a flexible mechanism for optimization, placing the burden of choosing or developing a particular algorithm on the user. This approach is motivated by the absence of known universal optimization techniques, and by the rapid advances in component technology which necessitate the continuous development of new optimization methods.

Although the Defcom feature of CIRCAL-2 could have been implemented by using the compiler of a common programming language, a special-purpose interpreter was chosen instead. The reasons behind this decision are the considerably smaller size of the interpreter; its ease of implementation; the desire to use the Defcom feature on-line without interrupting CIRCAL-2 execution; and the desire to make effective use of the CIRCAL-2 environment. The latter includes the flexibility of specifying statement forms familiar to the circuit designer; ease of reading and writing into the CIRCAL-2 data structure and Mode File; and the ability to utilize CIRCAL-2 functions and functionals within a Defcom.

A CIRCAL-2 Defcom consists of a sequence of statements which are specified by the user through the DESIGN command. A Defcom can operate on the following types of data:

(1) Network parameters (values of linear elements, and arguments of models, functions and functionals)

(2) Analysis and output parameters (analysis start, increment and end; plot start, increment and end)

(3) Values of fixed linear elements

(4) Variables saved by the last analysis.

In addition, any number of variables may be included as either arguments of, or local variables within a Defcom. The types of data that a Defcom can change are items (1) and (2) above, as well as Defcom arguments and local variables.

The algebraic expressions which may be used within Defcom statemenets are combinations of the above data types, or functions of expressions, connected by the ordinary arithmetic symbols (see Section IV). In addition, a Defcom may use Boolean expressions, derived from the binary relations $<$, $\leq$, $=$, $\neq$, $>$, $\geq$ on algebraic expressions, connected by the operators OR, AND, NOT, EXCLUSIVE-OR, and NOT-EXCLUSIVE-OR.

The Defcom statements are as follows:

(1) Assignment, of the form v = e

(2) Unconditional transfer to a specified label

(3) Conditional statements of the form

(a) <u>if</u> <boolean expression> <u>then</u> <statement>

(b) <u>if</u> <boolean expression> <u>then</u> $\langle\text{statement}_1\rangle$ <u>else</u> $\langle\text{statement}_2\rangle$

(4) Print

(5) Exit, which terminates execution of the Defcom

(6) Stop, which suspends execution of the Defcom

(7) Repeat, which performs a single re-analysis using the current values in the Mode File

(8) A Defcom name with arguments.

Observe that a Defcom may or may not return a value and thus may be used within an algebraic expression or within another Defcom. The "Exit" statement causes return to either the calling Defcom or to CIRCAL-2 command status. The "Stop" statement suspends execution of the Defcom and allows the user to specify on-line any desired Defcom statement. Such a statement is immediately executed, enabling the user to query or change values of the above data types before resuming execution of the suspended Defcom.

Finally, CIRCAL-2 incorporates certain <u>fixed-form</u>, higher-level Defcoms which perform common circuit-design tasks. An example of such a form is the following circuit-design-oriented Defcom:

<u>minimize</u> <E> <u>vary</u> <P> <u>step</u> <D> <u>while</u> <B>

which increases variable P from its present value, in steps of expression D, while Boolean expression B is True, in order to minimize expression E. Upon completion of its execution, this Defcom assigns to P that value which minimizes E, while B is true. As an example, this form could be used to minimize power dissipation in a transistor while the gain is held above a certain acceptable level.

XI USER CONVENIENCE FEATURES

Besides the convenience inherent in the interactive nature of CIRCAL-2, there are four specific features which form the user convenience package. They are: (1) express/slow mode option (2) interrupt (3) diagnostics, and (4) manual.

Since CIRCAL-2 is intended for use by a wide range of engineers with varying programming ability and CIRCAL-2 exposure, it is desirable to include optional features which increase or suppress the amount of interaction, thereby tailoring the system to the user's background. Such a feature is the slow/ express mode option, which was discussed in Section IX.

Experience obtained from earlier on-line circuit design programs indicates that on numerous occasions it is desirable to interrupt execution of the program. For example, this is the case when plotting a variable whose value becomes constant long before the end of the plot, or when it becomes evident that analysis is diverging. The interrupt feature implemented within CIRCAL-2 transfers to the next higher level in the program. Thus, if an interrupt is given during analysis or while plotting results, control is returned to command status, while if an interrupt is given during execution of an editing subcommand (Section III), control is returned to the editing level.

In an on-line system, especially in a rather complex one such as CIRCAL-2, it is necessary to have extensive diagnostics so that the source of an error can be rapidly determined and corrected. At present there are about 150 diagnostics stored on disk as short messages indicating the difficulty encountered. Isolation of the diagnostics from the program and their storage on disk rather than core, saves core space and allows convenient additions and modifications to be made to the list of existing diagnostics. The diagnostics are grouped in eight categories: (1) command (e.g. misspelled command or improper arguments), (2) editing (e.g. attempting to locate a nonexistent string of characters), (3) data-structure (e.g. use of nonexistent models or elements, improper number of nodes for a standard element), (4) function and functionals (e.g. syntactic errors in defining a function), (5) Mode File (e.g. requesting nonexistent analysis, incorrect computing dimension, incompatible plot and analysis parameters), (6) analysis (e.g. numerical overflow, use of elements not recognized by analysis program) (7) output (e.g. too many variables to be plotted), and (8) Defcom.

Errors occurring in CIRCAL-2 are either "fatal" or "nonfatal". Fatal errors return the user to command status while in the case of nonfatal errors, the system attempts to make a correction.

The MANUAL command allows a user to obtain all or part of the USER'S MANUAL whenever he is in command status. It may be used by relatively inexperienced users who are learning about the system, or by any user who wishes to clarify diagnostic messages, and take corrective action.

XII AN EXAMPLE

The design of a Colpitts oscillator is shown below as an example of CIRCAL-2 use. In this example, user input is in lower-case letters while computer output is in capitals; comments, in italics, provide further information.

After performing the operations necessary for establishing a connection with the CIRCAL-2 program (1) , the user requests information on the list file command (2) , and proceeds to use this command. He recognizes TRANS MODEL among the resulting list of files and requests to see its description (3). This is the model shown in Fig. 5. Note that the elements are specified in accordance with the format of Section V and associated Table 1. Satisfied with the model, the user proceeds to input the schematic of the Colpitts oscillator shown in the comment margin (4). The user specifies the model name incorrectly and enters the EDIT mode to correct his mistake. (The editing commands are explained in Section III.) Finally, the user checks the definition of the function, MAGNTD, which he intends to use later.

The user then requests the analysis of the oscillator by giving the OUTPUT command which requires that he supply all the necessary initialization data (Section IX) in slow mode (5). The program responds with a plot of the desired quantities: (1) the real part of the voltage at node 1, and (2) the magnitude of the voltage at node 1. The TIMER command is then used to determine the amount of time elapsed since the resumption of CIRCAL-2.

The user wishes to determine a value for the emitter resistance which causes the circuit to oscillate. It is known that oscillation occurs when the real part of the driving-point impedance becomes negative, and since, the circuit is driven by a positive current source, this is equivalent to the real part of the voltage at node 1 becoming negative. Noticing the trend from the first graph, the user repeats analysis with a higher value of emitter resistance (6). Note that all other Mode File parameters are left unchanged. The results indicate that this is the "right direction" since VR 1 is closer to zero.

The user then proceeds to automate his search for a value of emitter resistance which will cause a zero-crossover of VR 1 at 20 MHz, the desired oscillation frequency. He first checks the defined command called

IMPED which already exists in the system and whose action is precisely the desired search. The detail structure of IMPED is illustrated with a flow chart (7). Note that in this flow chart, REND is the specified maximum value of RESIS, and RINC is the stipulated increment of RESIS for a linear search. The IMPED command is executed (8) with the indicated arguments, and the value RESIS = 360 is returned as the answer, along with a graph of VR 1 versus frequency. The graph verifies that VR 1 crosses zero between 10 and 30 MHz. To establish the exact frequency at which the crossover occurs, the user asks for the data to be printed (9). Indeed, it is established from the resulting list that VR 1 (Column 2) becomes negative at exactly 20 MHz.

As a further check, the user calls a second frequency analysis routine[22], SFREQ, which uses sparse matrix techniques (10). The results are identical, hence, the user interrupts the printing (INT under Column 2) and terminates the entire CIRCAL-2 session by issuing the QUIT command (11).

```
1 MIT8B2: 18 USERS AT 06/18/70 1019.2, MAX = 30
  READY.

  login t378 stinger
  W 1019.4
  Password

   STANDBY LINE HAS BEEN ASSIGNED
   YOU HAVE    5166L SAVED
   T0378  5166 LOGGED IN  06/18/70 1019.7 FROM 800210
   LAST LOGOUT WAS  06/17/70 1856.0 FROM 800210

   CTSS BEING USED IS MIT8B2
  R 3.866+.250
```

Log in to CTSS by giving login command and password. System assigns a line and gives information about the last log in.

```
  resume circl2
  W 1022.4
  EXECUTION.
```

Request the CIRCAL-2 program to be resumed. COM- indicates CIRCAL-2 is ready for a command.

```
2 COM- manual listf

  THE FORMAT OF THE LISTF COMMAND IS
          (A) LISTF
          (B) LISTF NAME1 NAME2
          (C) LISTF   *   NAME2
          (D) LISTF NAME1   *
  WHERE (A) GIVES A LISTING OF ALL FILES, (B) LISTS ONLY THE FILE
  NAME1 NAME2 IF IT EXISTS, (C) LISTS ALL FILES WITH SECOND NAME NAME2,
  AND (D) LISTS ALL FILES WITH FIRST NAME NAME1.
```

Use the MANUAL command to obtain information about the LISTF command.

```
  COM- listf

  NAME1  NAME2  MODE LENGTH
  TRANS  MODEL   00     19
  MAGNTD FNCTN   10      7
  IMPED  DEFCOM  00     49
  OPTIM  DEFCOM  00     36
  RC     CKT     00      6
  COLP   MODFIL  00     13
  AMP    CKT     00     19
  DIFAMP CKT     00     19
```

List all files that have been created by the user in previous sessions with CIRCAL-2. Mode 10 for FNCTN file indicates function has been compiled. Length is in words.

```
3 COM- printf trans model

  XNODES 3 1 0 2
  ARCS 1 GM
  4 2 C2 6.E-12
  2 0 R3 17.E3
  1 4 R1 25.
  4 0 R2 150.
  4 0 C1 2.E-10
  2 0 J1 V(C1) GM
```

Print the contents of the file TRANS MODEL. Note the specification of the three external nodes and the single argument.

4

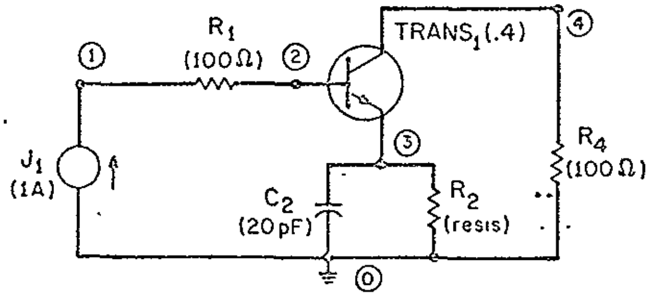
```
COM- design colpit ckt
INPUT
0 1 j1 1.
1 2 r1 100.
2 3 4 t1(.4)
3 0 r2 resis
3 0 c2 2.e-11
4 0 r4 100.

EDIT
1 t1
2 3 4 T1(.4)

c /t/trans/
2 3 4 TRANS1(.4)

f
*
```

Create the file COLPIT CKT containing the description of the circuit shown below. Note how the "locate" (l) and "change" (c) subcommands are used to change t1 to trans1.




```
COM- p magntd fnctn

MAGNTD(X, Y) = SQRT(X*X + Y*Y)
```

Print the contents of the file MAGNTD FNCTN. This function was defined and compiled during a previous CIRCAL-2 session.

5

```
COM- output colpit
TYPE ANALYSIS ROUTINE
freq
TYPE ANALYSIS INCREMENT, AI
ai = 2.e6
TYPE ANALYSIS START POINT, AS
as = 1.e7
TYPE ANALYSIS END POINT, AE
ae = 5.e7
TYPE INDEPENDENT VARIABLE, IV
iv = f
TYPE VARIABLES TO BE SAVED FOR OUTPUT
save 2 vi 1 vr 1
TYPE OUTPUT INCREMENT, PI

TYPE OUTPUT START POINT, PS

TYPE OUTPUT END POINT, PE

TYPE OUTPUT
plot 2 vr 1 magntd(vi 1 vr 1)
SPECIFY VALUE(S) FOR RESIS
resis = 300.
```

Use the OUTPUT command to analyze the circuit COLPIT. The mode file is filled in the slow mode, with the system requesting the value of each mode file parameter in turn. For those parameters which are not to be changed, a single carriage return is given. Note the request for the value of the parameter RESIS. Once the mode file has been filled, the circuit is analyzed and the requested plot of the analysis data is given.

```
RAPH OF
  1 VR 1
  2 MAGNTD(VI 1,VR 1)
VS F
VAR  SCALE
    FACTOR

1     1  1         3         5         7         9         11        13

2     2  2         7         12        17        22        27        32
         |----+----|----+----|----+----|----+----|----+----|----+----|

1.000E+07                             2                         1
                                2          1
                            2 1
                      1  2
                 1    2
2.000E+07     1     2
            1      2
           1      2
           1    2
           1    2
3.000E+07  1   2
           1  2
            12
             2
            2 1
4.000E+07   2  1
           2    1
           2     1
           2      1
          2         1
5.000E+07 2          1

OM- timer
OTAL RUNNING TIME =         8450 MS
```

Use the TIMER command to determine that 8.45 seconds of CPU time have been used so far.

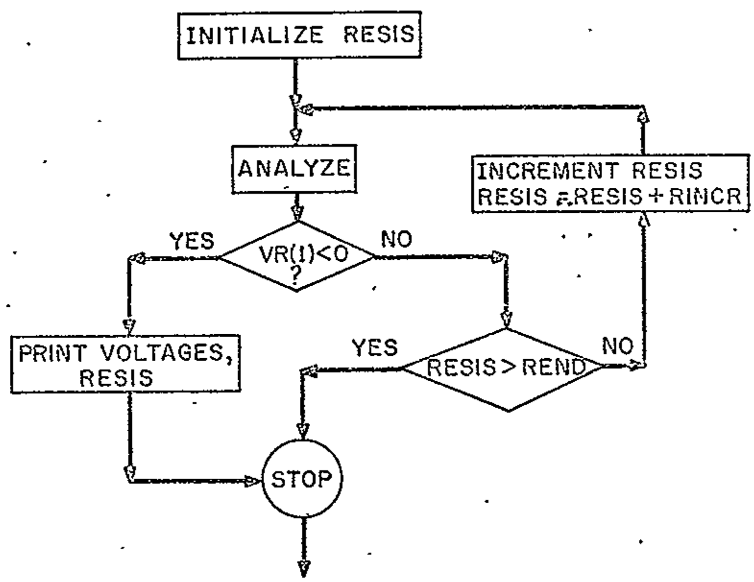
6
```
COM- repeat  resis = 330.  plot 1 vr 1
```

Use the REPEAT command to request re-analysis, specifying a different value for RESIS and a different output. Note that none of the other mode file parameters need be given - they retain their previous value.

```
GRAPH OF
  1 VR 1
VS F
VAR  SCALE
    FACTOR

1     1  0         2         4         6         8         10        12
         |----+----|----+----|----+----|----+----|----+----|----+----|

1.000E+07                                                    *
                                        *
                            *
                    *
                *
             *
           *
          *
          *
          *
3.000E+07  *
            *
             *
              *
               *
                *
                 *
                   *
                    *
                     *
5.000E+07              *
```



FLOW CHART FOR IMPED DEFCOM

7
```
COM- p imped defcom

ARGS 4 APT RSTART RINCR REND '
       RESIS = RSTART '
LOOP : REPEAT '
       IF VR 1 (APT) LES 0.
       THEN GOTO DONE
       ELSE IF RESIS GEQ REND
            THEN GOTO END '
       RESIS = RESIS + RINCR '
       GOTO LOOP '
DONE : PRINT RESIS '
END :  EXIT '
```

Print the contents of the file IMPED DEFCOM, a previously defined command. It will be used to determine the smallest value of RESIS for which the real part of the voltage at node 1 with respect to ground is negative, at analysis point APT.

8

```
COM- imped  2.e7  300.  20.  500.

RESIS= 3.599E+02

GRAPH OF
  1 VR 1
VS F
VAR  SCALE
    FACTOR

1     0  -8         2        12        22        32        42        52
          |----+----|----+----|----+----|----+----|----+----|----+----|----+----|

1.000E+07
                                                                 *
                                          *
                            *
                    *
              *
            *
          *
          *
           *
3.000E+07    *
              *
                 *
                   *
                      *
                         *
                            *
                               *
                                  *
                                     *
5.000E+07                               *
```

Use the defined command with APT=20MHz, RSTART=300Ω, RINCR=20Ω, and REND=500Ω. The circuit is analyzed four times before the voltage goes negative, at which point RESIS=360Ω. The final analysis data are plotted.

(9)
```
COM- r print 3 vl 1 vr 1 magntd(vl 1 vr 1)  pe = 3.e7

VALUES FOR
  1 VI 1
  2 VR 1
  3 MAGNTD(VI 1,VR 1)
VS F

     IV              1              2              3
  1.000E+07     -1.696E+03     8.721E+01      1.698E+03
  1.199E+07     -1.406E+03     4.721E+01      1.406E+03
  1.399E+07     -1.196E+03     2.408E+01      1.196E+03
  1.599E+07     -1.038E+03     1.010E+01      1.038E+03
  1.799E+07     -9.135E+02     1.550E+00      9.135E+02
  2.000E+07     -8.131E+02    -3.566E+00      8.131E+02
  2.199E+07     -7.304E+02    -6.384E+00      7.304E+02
  2.399E+07     -6.609E+02    -7.598E+00      6.609E+02
  2.599E+07     -6.017E+02    -7.661E+00      6.018E+02
  2.799E+07     -5.508E+02    -6.877E+00      5.508E+02
  3.000E+07     -5.064E+02    -5.460E+00      5.064E+02
```

Request the analysis data to be printed, ending at 30MHz. Re-analysis is not required since the data from the previous analysis has been saved.

(10)
```
COM- r sfreq

VALUES FOR
  1 VI 1
  2 VR 1
  3 MAGNTD(VI 1,VR 1)
VS F

     IV              1              2              3
  1.000E+07     -1.696E+03     8.721E+01      1.698E+03
  1.199E+07     -1.406E+03     4.721E+01      1.406E+03
  1.399E+07     -1.196E+03     2.408E+01      1.196E+03
  1.599E+07     -1.038E+03     1.010E+01      1.038E+03
  1.799E+07     -9.135E+02     1.550E+00      9.135E+02
  2.000E+07     -8.131E+02    -3.567E+00      8.131E+02
  2.199E+07     -7.304E+02    -6.384E+00      7.304E+02
  2.399E+07     -6.609E+02      INT. 1
```

To check the data, the circuit is re-analyzed with a different frequency analysis routine which uses sparse matrix techniques. Observing that the two sets of data agree, the user interrupts the printing and the system returns to command status.

(11)
```
COM- quit
R 31.483+28.633
```

Use the QUIT command to leave CIRCAL-2. This session with CIRCAL-2 used 31.483 seconds of CPU time and 28.633 seconds of swap time.

XIII CONCLUSIONS

Since CIRCAL-2 has been used primarily by the people who developed it, it is difficult, at this time, to arrive at objective conclusions on the effectiveness of the program in a commercial environment. On the basis of program use to date, however, the following conclusions may be drawn:

The multiple analysis approach is indeed practical and has been found effective in the design of new analysis techniques. Specifically, two transient and two frequency analysis routines, and one D.C. analysis method based on recursive decomposition have been, or are in the process of being implemented. Without exception, the development of these routines was conducted without detailed knowledge of the overhead portions of CIRCAL-2, such as the compilation of functions and functionals, the file system, and the output processor. The form of the network data structure and the mode file were found to considerably reduce the effort involved in writing new analysis routines for CIRCAL-2.

The common elements have, so far, posed little restriction on the development of analysis techniques. However, it has been observed that a need does exist to allow analysis routine developers to define their own types of elements, such as diodes, which are then treated in a special manner in order to simplify the analysis task. This has been incorporated in the most recent version of CIRCAL-2.

It has been found that functions and functionals can be effectively shared by various parts of the system. For example, the same function has been used to describe an element, as part of a user-defined optimization routine, and for the output of analysis results.

The use of the Mode File for incremental modification substantially reduces the amount of interaction time, as compared to CIRCAL-1.

The power of the defined command and functional features has not been fully exploited because of the concentration on program development rather than on program use. It is therefore premature to assess the utility of these features at this time.

CIRCAL-2 was originally developed on the Project MAC[4] IBM 7094 computer, modified for time sharing. CIRCAL-2 has recently been converted* for both batch and on-line IBM-360 use and for acceptance of FORTRAN analysis routines. In both cases the source language was AED.[18]

* By Softech Inc., 391 Totten Pond Road, Waltham, Mass. 02154, which is marketing CIRCAL-2.

XIV REFERENCES FOR SECTION B

1. M.L. Dertouzos and C.W. Therrien, "CIRCAL: On-Line Analysis of Electronic Networks", MIT Electronic Systems Lab, Cambridge, Mass., Rept. ESL-248, OCtober 1965.

2. M.L. Dertouzos, "CIRCAL: On-Line Circuit Design," Proc. IEEE, vol. 55, pp. 637-654, May 1967.

3. C.W. Therrien, "A digital computer simulation of electrical networks," S.M thesis, Dept. of Electrical Engineering, MIT, Cambridge, Mass., June 1965.

4. R.M. Fano, "The MAC System: The Computer utility approach", IEEE Spectrum, vol.2, pp. 56-64, January 1965.

5. J. Katzenelson, "AEDNET: A Simulator for nonlinear networks", Proc. IEEE, vol. 54, pp. 1536-1552, November 1966.

6. D.S. Evans and J. Katzenelson, 'Data structure and man-machine communication for network problems," Proc. IEEE, vol. 55, pp. 1135-1144, July 1967.

7. H.C. So, 'OLCA: An On-Line circuit analysis system", Proc. IEEE, vol 55 No. 11, November, 1967.

8. M.L. Dertouzos, "An introduction to on-line circuit design", Proc. IEEE, vol. 55, no. 11, pp. 1961-1971, November, 1967.

9. P.A. Crisman, (editor) "The compatible Time-sharing System", A Programmer's Guide, Cambridge, Mass., MIT Press, 1965.

10. D.L. Isaman, 'On-Line creation of functions for computer-aided design, S.M. thesis, Dept. of Electrical Engineering, MIT. Cambridge, Mass., February, 1970.

11. J.R. Stinger and M.L. Dertouzos, 'CIRCAL-2-User's Manual,' MIT Electronic Systems Lab, Cambridge, Mass., Rept. ESL-R-381, May 1969.

12. "1620 electronic circuit analysis program (ECAP).' IBM Corporation, Data Processing Div., White Plains, N.Y., Application Program 1620-EE-02x.

13. A.F. Malmberg, F.L. Cornwell, and F.N. Hofer, "NET-1 network analysis program,' Los Alamos Scientific Lab., Los Alamos, N. Mex., Rept. LA-3119, 1964.

14. L.D. Milliman, W.A. Massena, R.H. Dickhaut, and A.C. Mong, "CIRCUS, a digital computer program for transient analysis of electronic circuits, program manual," U.S. Army Materiel Command, Harry Diamond Laboratories, Washington, D.C., January, 1967.

15. H.W. Mathers, S.R. Sedore, and J.R. Sents, "Automated digital computer program for determining responses of electronic circuits to transient nuclear radiation (SCEPTRE)," vol. I, IBM Space Guidance Center, Owego, N.Y., IBM File 66-928-611, February 1967.

16. "MISSAP III, general information manual, electronic circuit analysis," Michigan State University, East Lansing, Michigan, July, 1966.

17. D.T. Ross, "Introduction to software engineering with the AED-0 language," MIT Electronic Systems Lab., Cambridge, Mass., Rept. ESL-R-405, October 1969.

18. C.L. Feldman, D.T. Ross, and J.E. Rodriguez, "AED-0 Programmer's Guide," MIT Electronic Systems Lab., Cambridge, mass., Rept. ESL-R-406, January 1970.

19. H.L. Graham, "Recursive graph decomposition: an approach to computer analysis of networks," PhD thesis, Dept. of Electrical Engineering, MIT, Cambridge, Mass., June 1969.

20. M.L. Dertouzos, "Computer analysis of nonlinear networks by recursive decomposition," Proc. 2nd Biennial Cornell Electrical Engineering Conference, pp. 57-67, August 1969.

21. C.L. Searle, et.al., Elementary Circuit Properties of Transistors, SEEC, vol.3, Wiley, N.Y., 1964. p. 84.

22. K.A. Van Bree, "Frequency analysis of linear circuits using sparse matrix techniques," S.M. thesis, Dept. of Electrical Engineering, MIT, Cambridge, Mass., August, 1969.

C. PUBLICATIONS OF THE PROJECT

Reports

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-225, December, 1964.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-245, June, 1965.

Dertouzos, M. L., and Therrien, C.W., "CIRCAL: On-Line Analysis of Electronic Networks," Report ESL-R-248, December, 1965.

Dertouzos, M. L., and Santos, P.J., Jr., "CADD: On-Line Synthesis of Logic Circuits," Report ESL-R-253, December, 1965.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-256, December, 1965.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-274, June, 1966.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-298, January, 1967.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-322, September, 1967.

"Computer-Aided Electronic Circuit Design," Part I, Status Report ESL-SR-337, January, 1968.

"Computer-Aided Electronic Circuit Design," Status Report, ESL-SR-363, September, 1968.

"Computer-Aided Electronic Circuit Design," Status Report, ESL-SR-379, April, 1969.

"Computer-Aided Electronic Circuit Design," Final Report, ESL-FR-436, September, 1970.

II. Theses

Santos, P., "CADD, A Computer-Aided Digital Design Program," Master of Science Thesis, Department of Electrical Engineering, MIT. June 1965.

Therrien, C.W., "Digital-Computer Simulation for Electrical Networks," Master of Science Thesis, Department of Electrical Engineering, June, 1965.

II. Theses (Cont'd)

Fluhr, Z.C., "Single-Threshold Element Realizability by Minimization," Master of Science Thesis, Department of Electrical Engineering, August, 1965.

Olansky, K.J., "A Low-Cost Teletype-Operated Graphical Display," Master of Science Thesis, Department of Electrical Engineering, August, 1965.

Gertz, J. L., "A Graphical Input-Output Program for Digital System Simulation," Master of Science Thesis, Department of Electrical Engineering, June, 1966.

Graham, H.L., "A Hybrid Graphical Display Technique," Master of Science Thesis, Department of Electrical Engineering, June, 1966.

Meltzer, J.R., "CIRCAL: An Input for Nonlinear Elements," Master of Science Thesis, Department of Electrical Engineering, June, 1966.

Taubman, C.N., "Computer Analysis of Electrical Analog Distribution Systems," Master of Science Thesis, Department of Electrical Engineering, June, 1966.

Walpert, S.A., "An Output Program for Representing Electrical Signals," Master of Science Thesis, Department of Electrical Engineering, June, 1966.

Edelberg, M., "A Dual Approach to Threshold Decomposition of Boolean Functions," Master of Science Thesis, Department of Electrical Engineering, June, 1967.

Willems, J.D., "Synthesis of Logical Functions with Restricted Threshold Elements," Electrical Engineer Thesis, Department of Electrical Engineering, June, 1967.

Smith, T., "Nesting of Networks for Computer-Aided Circuit Design," Bachelor of Science Thesis, Department of Electrical Engineering, June, 1967.

Kaliski, M.E., and Polzen, K.P., "LOTUS, On-Line Simulation of Block Diagram Systems," Master of Science joint thesis, Department of Electrical Engineering, MIT, January, 1968

Stinger, J.R., "A General Data Structure for On-Line Circuit Design," Master of Science Thesis, Department of Electrical Engineering, MIT, January, 1968.

Dvorak, A.A., "An Input-Output Program for Electronic Circuits Using a CRT," Bachelor of Science Thesis, Department of Electrical Engineering, MIT, March, 1968.

II. Theses (Cont'd)

Therrien, C.W., "Tearing of Networks," Doctoral Thesis, Department of Electrical Engineering, MIT, June, 1969.

Van Bree, K.A., "Frequency Analysis of Linear Circuits Using Sparse Matrix Techniques," Master of Science Thesis, Department of Electrical Engineering, MIT, September, 1969.

Isaman, D.L., "On-Line Creation of Functions for Computer-Aided Circuit Design," Master of Science Thesis, Department of Electrical Engineering, MIT, February, 1970.

III. Technical Papers and Conference Participation

Reintjes, J.R., and Dertouzos, M.L., "Computer-Aided Design of Electronic Circuits," WINCON Conference, February, 1966, Los Angeles, California.

Reintjes, J.F., "The Role of Computers in Modern Design Technology," Conference on Computer-Aided Design, University of Wisconsin, May 3-4, 1966.

Dertouzos, M.L., and Graham, H.L., "A Parametric Graphical Display Technique for On-Line Use," presented at Fall Joint Computer Conference on November 8, 1966. Published in the Conference Proceedings of the FJCC.

Therrien, C.W., and Dertouzos, M.L., "CIRCAL: On-Line Design of Electronic Circuits," presented at the NEREM Show and published in NEREM Record, November 9, 1966.

Notes: (1) Professor Dertouzos was Chairman of the Computer-Aided Electronic Circuit Design Session at the NEREM 1966 Conference, Boston, Massachusetts.

(2) CIRCAL-I was used from Munich, Germany via TELEX, June, 1966, in connection with a series of ten lectures by Professor Dertouzos at Siemens, Halske.

Katzenelson, J., "AEDNET: A Simulator for Nonlinear Networks," Proceedings of the IEEE, Vol. 54, No. 11, November, 1966, pp. 1536-1552.

Therrien, C.W., presentation on CIRCAL at NYU Conference on "Network Analysis by Computer Symposium," New York University, January, 1967.

III. Technical Papers and Conference Participation (Cont'd)

Dertouzos, M. L., and Graham, H. L., "A Parametric Graphical Display Technique for On-Line Use," MIT Project MAC Seminar, May 19, 1966.

Dertouzos, M. L., "PHASEPLOT: An On-Line Graphical Display Technique," IEEE Transactions on Electronic Computers, Vol. EC-16, No. 2, April 1967, pp. 203-209.

Dertouzos, M. L., and Fluhr, Z. C., "Minimization and Convexity in Threshold Logic," Seventh Annual Symposium on Switching Circuit Theory and Logical Design, Berkeley, California, 1966; IEEE Transactions on Electronic Computers, Vol. EC-16, No. 2, April 1967, pp. 212-215.

Dertouzos, M. L., Panelist in panel discussion, "On-Line Versus Batch," held at the NASA Computer-Aided Circuit Design Seminar, April 11-12, 1967, Cambridge, Massachusetts.

Dertouzos, M. L., "CIRCAL: On-Line Circuit Design," Proceedings of the IEEE, Vol. 55, No. 5, May, 1967, pp. 637-654.

Dertouzos, M. L., Co-chairman of and Lecturer at MIT Summer Course 6.56S on "On-Line Circuit Design," MIT July 6 - July 13, 1967. (One set of Proceedings issued at that course.)

Evans, D. S., and Katzenelson, J., "Data Structure and Man-Machine Communications Problems," Proceedings of the IEEE, Vol. 55, No. 7, July, 1967, pp. 1135-1144.

Dertouzos, M. L., Co-chairman of and Lecturer at Industrial Liaison Symposium on "Computer-Aided Circuit Design," MIT, October 3, 1967.

Dertouzos, M. L., "An Introduction to On-Line Circuit Design," Proceedings of the IEEE, Vol. 55, No. 11, November, 1967, pp. 1961-1971; also Proceedings of the Fifth Allerton Conference on Circuit and System Theory, October 4-6, 1967 (Invited Paper).

Dertouzos, M. L., Chairman of Session "Computer-Aided Circuit Design; A Critical Appraisal," NEREM 1967, Boston, November 1967.

Dertouzos, M. L., Talks and Demonstration on On-Line System and Circuit Simulation to:

a. EE Department, Catholic University of America, Washington, D.C., November 10, 1967.

b. IEEE Computer Group, Boston, October 11, 1967.

III. Technical Papers and Conference Participation (Cont'd)

Dertouzos, M. L., "Panel Discussion on Computer-Aided Design," Proceedings of the IEEE, Vol. 55, No. 11, November, 1967, pp. 1777-1778.

M. L. Dertouzos, "Man-Machine Interaction in Circuit Design", Proceedings of the Hawaii International Conference on System Sciences, University of Hawaii, Honolulu, Hawaii, January 29-31, 1968.

M. L. Dertouzos, Offered Special Summer Course on MIT subject 6.001, "Elements, Systems, and Computation", sponsored by the COSINE Committee, Princeton University, Princeton, New Jersey, June 18-28, 1968.

M. L. Dertouzos, A series of Lectures on Computer-Aided Design at Siemens A.G., Munich, Germany, July 15-19, 1968.

M. L. Dertouzos, Lecture, "Computer-Aided Circuit Design," MIT-Technical University of Berlin Joint Summer Conference on "The Computer in the University," Berlin, Germany, July 22-August 2, 1968.

M. L. Dertouzos, Chairman, Computer Graphics Session and Panelist in two panel discussions, MIT-Technical University of Berlin Joint Summer Conference on "The Computer in the University", Berlin, Germany, July 22 - August 2, 1968.

J. A. Narud, M. L. Dertouzos, G. P. Jessel, "Computer-Aided Analysis for Integrated Circuits", INVITED PAPER, Proceedings of the 1968 IEEE International Symposium on Circuit Theory, Miami Beach, Florida, December 2 - 4, 1968. (Also to appear in IEEE Spectrum, "Computer-Aided Design for Integrated Circuits", December, 1970.)

M. L. Dertouzos, Panelist, "Computer-Aided Circuit Design," IEEE International Solid State Circuits Conference, Philadelphia, Pennsylvania, February 20, 1969.

M. L. Dertouzos, Panelist, "Design Automation Session", 1969 Spring-Joint Computer Conference, Boston, Massachusetts, May 14 -16, 1969.

J.R. Stinger, M.L. Dertouzos, "CIRCAL-2: A General-Purpose On-Line Circuit-Design Program User's Manual", Status Report, ESL-R-381, MIT, May 1969.

M. L. Dertouzos, "Computer Analysis of Nonlinear Networks by Recursive Decomposition", Proceedings, Second Biennial Cornell Electrical Engineering Conference on Computerized Electronics, Vol. 2, August 1969.

III. Technical Papers and Conference Participation (Cont'd)

M. L. Dertouzos, "Educational Uses of On-Line Circuit Design", IEEE Transactions on Education, Vol. E-12, No. 3, September 1969.

M. L. Dertouzos, "Character Generation from Resistive Storage of Time Derivatives", AFIPS Conference Proceedings (Fall Joint Computer Conference), Vol. 35, November 1969.

Jessel, G. P., and Stinger, J. R., "CIRCAL-2 - A General Approach to On-Line Circuit Analysis", NEREM 1969, Boston, Mass., November 5, 1969.

M. L. Dertouzos, "Computer Science Education in the Soviet Union", IEEE SPECTRUM, Vol. 6, No. 12, December 1969.

IV. Patents, Movies

M. L. Dertouzos, H. L. Graham, U. S. Patent No. 3,449,721 on Display System.

CIRCAL: Computer-Aided Electronic Circuit Design, January, 1966.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SCIENCE AND ARTS

1861

MENS ET MANUS