INTERFACING INTERACTIVE CIRCUIT SIMULATION WITH STANDARD GRAPHICS FACILITIES

P. Gillard, W.M. Zuberek
Department of Computer Science, Memorial University of Newfoundland
St. John's, Newfoundland, Canada A1C 5S7

Abstract
One of the most popular and most powerful circuit simulators, the SPICE-2G program from University of California, Berkeley, is rather inefficient in interactive applications because of its "batch-oriented" structure. SPICE-PAC is a package (or a set) of subroutines which is functionally equivalent to the SPICE-2G circuit simulator, i.e., it accepts the same circuit description and performs all the analyses which are available in the SPICE-2G programs, but also provides several extensions, e.g., circuit variables, parameterized subcircuit expansion, and an interface to libraries of standard modules. Two immediate applications of the package are: (1) interactive circuit simulation in which an interactive driver controls the simulation subroutines according to user commands, and (2) circuit optimization in which an optimization package is interfaced with the simulation package by user-supplied subroutines that evaluate objective functions, constraints, etc. Recently, the package has been extended by an interface to graphics facilities. An implementation of this interface, compatible with both the CORE and GKS systems, is described, and some simple examples are given as an illustration.

1. INTRODUCTION

Computer-aided circuit analysis, or circuit simulation, has become a widely accepted tool in integrated circuit design [6,10,11]. Using this method, circuit designers can easily explore the effects of different designs on the circuit performance. The SPICE-2 program [8,14] developed at the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, has become one of the most popular "second-generation" circuit simulators.

The SPICE-2 program execution consists of two basic phases [8]. The first phase reads all the input data (i.e., the circuit description and parameters of required analyses), while the second phase performs all the simulations and prints the results. The consequence of such a program organization is that even a minor change in any of the element descriptions or parameter values requires a new, independent run of the simulator. To provide flexible and efficient analyses of numerous variants of the same circuit (which is often required to "optimize" or "center" the initial design), a new structure of the circuit simulator is needed, in which different analyses (for the same circuit) can be performed selectively, and in which there is an access to internal representation of circuit elements in order to modify their values. SPICE-PAC is a package (or a set) of subroutines which is functionally equivalent to the SPICE-2G.6 circuit simulation program, i.e., it not only accepts the same circuit description and performs all the analyses which are available in the SPICE 2G programs, but also provides an access to internal values of circuit elements, dynamic definitions of parameters and outputs, a hierarchical naming scheme for subcircuit elements, parameterized subcircuit expansions, and an interface to libraries of standard modules. Two immediate applications of the package are: (1) interactive circuit simulation, in which users selectively perform circuit analyses and modify circuit elements to satisfy design requirements, and (2) circuit optimization, in which an optimization package is interfaced with the simulation package by user-supplied subroutines that evaluate objective functions, constraints, etc.

The advantages of this very flexible structure of the simulation package can, however, be seriously degraded by a traditional, numerical representation of results. This is especially acute in interactive applications where tedious studying of long columns of (very accurate) numerical results is usually much more time consuming and far less useful than simple comparisons of (less accurate) graphical data. Therefore the SPICE-PAC package has been extended by an interface to graphics facilities, and since the CORE graphics system has been more widely implemented than other graphics standards in North America [4,7,13], the CORE system was chosen rather than the GKS standard [2,15] as the "basic" level of reference.

The CORE system is a package of subroutines which defines an abstract, device-independent set of graphics capabilities. It was designed for functional completeness so that any graphics function is either included within the system or can be easily built on top of CORE system routines. The general functional capabilities found in the system are of six types:

1) output primitives which are lines, character strings, polygons and markers,
2) output primitive attributes such as line color, style and intensity, character font, etc.,
3) segments which are disjoint collections of output primitives, and which can be created, deleted, transformed, blinked, made temporarily invisible, and picked using an input device,
4) viewing capabilities to create an image of an object in a viewport on a view surface,
5) operator interaction,
6) overall control to select the surface, establish default attributes, etc.

For circuit simulation applications, a small subset of these functions is satisfactory, and a "basic" subset can be selected in such a way, that only a very simple conversion is required to transform it into a subset of the GKS standard. In fact, the minimal subsets proposed for both CORE and GKS are entirely adequate for this application.

This paper discusses the general idea of interfacing circuit simulation packages with the CORE and GKS graphics systems, describes some "higher-level" functions defined within the GRAPH-PAC set of subroutines, and shows some examples of graphical results of circuit simulations and optimizations.

2. SPICE-PAC 2G6a

SPICE-PAC version 2G6a is a package of simulation subroutines obtained by redesigning the SPICE 2G.6 simulation program. The package provides:

(a) the same circuit descriptions as for the SPICE 2G programs (in fact, there are a few minor differences but still the SPICE input language is accepted by the SPICE-PAC package),
(b) all analyses available in the SPICE 2G programs; since all the analyses are performed "on demand" by calling appropriate subroutines of the package, there is no restriction on the ordering or number of analyses performed within a single simulation session,
(c) access to "circuit variables" as required in interactive simulation and circuit optimization; circuit variables are those circuit elements which can be modified during a simulation session; circuit variables can be defined at the main circuit level as well as in subcircuits,
(d) dynamic declarations of parameters and output variables for all analyses; output variables can be indicated at the main circuit level as well as in subcircuits,
(e) parameterized subcircuit calls; subcircuit element definitions can be redefined by parameters included in subcircuit calls,
(f) an interface to libraries of standard modules; standard modules are in the form of subcircuits stored in individual files within a file system, and they can be accessed by (parameterized) module calls.

SPICE-PAC contains 25 "main" (or interfacing) subroutines but does not provide the "main program" which must be suplied by the user to "drive" the subroutines, i.e., to call the subroutines which read a circuit description, define circuit variables, perform analyses, etc., as required by a particular application.

Two immediate examples of SPICE-PAC applications are interactive simulation and circuit optimization.

In interactive circuit simulation, the driving routine mainly handles communication with the user, i.e., it enters user commands, converts them into corresponding sequences of SPICE-PAC subroutine calls, and displays (in numerical or graphical form) the results. The complexity of the driver is directly related to the sophistication of interaction. In typical applications interaction is rather simple, and it can use, for example, commands similar to the SPICE input language. A "convenient" representation of results appears to be a more troublesome problem, and a graphical form is probably the easiest to understand and the most efficient and compact way to present large amounts of data. Since some selected data (e.g., the final solution) may be required in numerical form also (for further processing), the graphical representation should be "backed-up" by a convenient way of storing original numerical results obtained from circuit simulations.

Because the output from SPICE-PAC is very likely to be used for applications other than display, it is of no real advantage to store the results as graphical information in a metafile. This application is somewhat unique in that the computational effort involved in the simulation is normally very much more than that required to provide the graphical display.

In circuit optimization, the driving routine has to control at least two packages, the optimization and the circuit simulation one. In many cases, when indirect communication is used for interfacing circuit simulation with optimization, the packages are hierarchically structured, and then the whole optimization process is controlled by the optimization algorithm used. This also means that the user can influence the optimization process only by selection of the starting point and (some of) the optimization parameters. More flexible (but also more difficult) solution is to use reverse communication

[9] in which case user routines (or user - in an interactive way) can control the optimization process at the "step" level, and can adjust the parameters even "during" the optimization process, or can interrupt an "unsuccessful" iteration to change the starting point, or can run several optimizations "concurrently" to compare the results and select the "best" ones. In all such cases a graphical "trace" of the optimization progress is practically the only representation of (usually multidimensional) information which can be used by users in "real-time" to control and direct the optimization. Some very simple examples of such traces are shown in section 5.

3. GRAPHICS PACKAGES

The graphics tools required for an interface to SPICE-PAC are relatively unsophisticated. Normally, several circuit parameters are to be varied in a given simulation or optimization session, and these (numeric) inputs do not readily lend themselves to graphical input. As well, most applications require immediate plotting of results, and do not require the use of retained segments. Consequently, a graphics system which provides a simple, two dimensional plotting capability with no graphical input and which supports only temporary segments is satisfactory. A metafile capability is not required, because the results of the analysis will be saved for other applications, and can readily be replotted. In CORE, a system providing "basic", "no input", "2D" capabilities is quite adequate. In the GKS system, output level 0 (basic output), input level a (no input) is adequate; in fact, the "minimal GKS" implementation described by Simons [12] is sufficient.

Because there are basic differences between the CORE and GKS graphics systems [15] (e. g., in the GKS system there is no concept of "current position"), an implementation of the GRAPH-PAC to GKS interface may behave slightly differently from the case of an interface to CORE. In particular, it may be necessary to buffer the output of some of the plot functions, and output may occur later than in the CORE implementation. Additionally, for text positioning, it may be necessary to store the "current position". Relative coordinates are also not provided in GKS, but could easily be simulated.

Table 1 shows the primitive functions required to implement the present version of the graphics driver for SPICE-PAC in both the CORE and GKS graphics system. Also included are the function names for a commercial FORTRAN implementation of the CORE graphics standard, DI-3000 [1].

In addition to the graphics functions listed in Table 1, a number of "second level" functions were defined to assist the user in defining the display region. In certain circuit analyses, the designer may want to use several different graphical scales, such as linear, log-linear, log-log to display the full range of data. Therefore it is convenient to have a scale function which allows the plotting of "scaled" data from "real" input data. Therefore, plotting functions were defined which automatically scale the data to correspond to a predefined set of scale functions. Normally, all plot commands position plotted points after applying the appropriate "scale" functions to both the x- and y-coordinates of the data.

A number of other functions were defined to allow axes to be drawn through a point, to place labeled ticks arbitrarily at a point (say, on an axis, to provide an arbitrary axis labeling). Some of these functions, defined in terms of the functions available in the lowest level graphics package, are the following:

```
GPSCALE(XMIN,XMAX,XFN,YMIN,YMAX,YFN)
DOUBLE PRECISION XMIN,XMAX,YMIN,YMAX
INTEGER XFN,YFN
```

GPSCALE defines a transformation function for each of the x and y coordinates, using the integer arguments XFN and YFN. A window is then created, using the scaled values of XMIN, XMAX, YMIN and YMAX. Presently, the transformation functions are:

-2	natural log
-1	decibel (10 * log10)
0	log10
1	linear
n	polynomial, x**n or y**n

there are also generalized DRAW and MOVE functions which draw visible or invisible lines from the current position to the points determined by applying the appropriate scale functions to their arguments.

```
GPAXIS(XO,YO,XTIC,YTIC,XLABEL,YLABEL)
DOUBLE PRECISION XO,YO,XTIC,YTIC
CHARACTER*(*) XLABEL,YLABEL
```

GPAXIS draws a set of axes through the point (XO,YO) with ticks spaced a distance XTIC and YTIC on the x and y axes, respectively. The ticks are labeled automatically, and the labels XLABEL and YLABEL are written on the x and y axes. The ticks are uniformly spaced in the "scaled" coordinates, and labeled with "real" coordinate values. All labeling is done to the left and/or the bottom of the plotting area, and the viewport is redefined to be the region included in the plot area, which excludes the possibility of drawing lines in the text areas.

Table 1. Primitive graphics functions for GRAPH-PAC

CORE	GKS	GKS (FORTRAN)	DI-3000
1. initialization			
initialize_core	OPEN GKS	GOPKS	JBEGIN
initialize_view_surface	OPEN WORKSTATION	GOPWK	JDINIT
select_view_surface	ACTIVATE WORKSTATION	GACWR	JDEVON
create_temporary_segment			JOPEN
new_frame	CLEAR WORKSTATION	GCLRWK	JFRAME
2. viewing			
set_window	SET WINDOW	GSWN	JWINDO
set_viewport_2	SET VIEWPORT	GSUP	JVPORT
inquire_window	INQUIRE NORMALIZATION	GQNT	
inquire_viewport	TRANSFORMATION		
3. drawing primitives			
line_abs_2	POLYLINE	GPL	JDRAW
move_abs_2			JMOVE
move_rel_2			JRMOVE
marker_abs_2	POLYMARKER	GPM	JMARK
4. drawing attributes			
set_linestyle	SET LINETYPE	GSLN	JLSTYL
set_marker_symbol	SET MARKER TYPE	GSMK	JCMARK
5. text primitives			
text	TEXT	GTX	J3STRG
6. text attributes			
set_font	SET TEXT FONT	GSTXFP	JFONT
set_charsize	SET CHARACTER HEIGHT	GSCHH	JSIZE
set_charjust	SET TEXT ALIGNMENT	GSTXAL	JJUST
set_charpath	SET TEXT PATH	GSTXP	JPATH
set_charup_2	SET CHARACTER UP VECTOR	GSCHUP	
inquire_text_extent_2	INQUIRE TEXT EXTENT	GQTXX	
7. termination			
close_temporary_segment			JCLOSE
deselect_view_surface	DEACTIVATE WORKSTATION	GDAWK	JDEVOF
terminate_view_surface	CLOSE WORKSTATION	GCLWK	JDEND
terminate_core	CLOSE GKS	GCLKS	JEND

GPTICK(X,Y,DIR,LABEL)
DOUBLE PRECISION X,Y
INTEGER DIR
CHARACTER*(*) LABEL

GPTICK places a tick at point (X,Y) in direction DIR (1- right, 2- down, 3- left, 4- up) and writes the character string LABEL beside the tick.

GPBOX(XL,YB,XR,YT)
DOUBLE PRECISION XL,YB,XR,YT

GPBOX draws a box with lower left coordinate (XL,YB) and upper right coordinate (XR,YT).

4. IMPLEMENTATION ISSUES

One of the most significant differences between CORE and GKS is the presence of language bindings in GKS. For languages like FORTRAN, C, PASCAL or ADA, subroutine names, calling sequences and data types are given in the standard specification. This means that programs should require essentially no changes to move among different environments supporting GKS. In CORE, no standard syntax was given for any programming language, and implementors strayed to varying extents from the CORE specification. This is not very comforting to users who try to develop portable application programs.

To overcome these differences, a "2-level" solution has been attempted in GRAPH-PAC. The first (lower) level is a set of subroutines which provide an interface between a "standard" graphics package (CORE, GKS, or another one) and the second level. The second level provides a "user" interface, i.e., it is a set of subroutines which implement "user oriented" functions in terms of primitives supported by the first level. In the case of moving from one graphics environment to another (e.g., from CORE to GKS), only the lower-level interface must be replaced by a new, appropriate set of "binding" subroutines which are usually rather simple, and in many cases perform just a renaming or reordering of calling sequences.

Moreover, the lack of standardization of calling sequences in CORE may result in difficulties in accessing the CORE implementation from different languages. For example, if parameter passing mechanisms are not the same for different higher-level languages, a "low-level" routines must be used to adjust the parameter representations. In our implementation, the CORE graphics system was developed for a C programming environment, and does not conform to FORTRAN parameter passing conventions. Therefore it was necessary to implement a part of the lower-level GRAPH-PAC interface in the C language.

Also, since GRAPH-PAC must be linked with several other packages, some of which are very complex and sophisticated (e.g., SPICE-PAC and optimization packages, with hundreds of internal subroutines), strict naming conventions must be imposed to avoid naming conflicts among globally defined objects. Therefore it was assumed that within packages all global names begin with the same prefix. For GRAPH-PAC the prefix is "GP", for SPICE-PAC it is "SP", for optimization packages "Wxyz", where "xyz" identifies the package (DI-3000 has a similar convention and all subroutine names begin with "J"; for the GKS FORTRAN binding, each subprogram name begins with "G").

Presently, GRAPH-PAC is implemented on VAX-11 systems under the 4.2-BSD-UNIX operating system, and will shortly be ported to the VAX-11/VMS environment. Together with SPICE-PAC and optimization libraries, it is a part of CAD software being developed for design and verification of electronic circuits.

5. EXAMPLES

The first example is an interactive simulation of a differential amplifier in which case a DC transfer curve [14] is analyzed, and a linear region, symmetrical with respect to zero input voltage, should be obtained. Fig.1 shows 3 pairs of transfer characteristics (denoted by V(3) and V(5), V(3)a and V(5)a, and V(3)b and V(5)b, respectively) obtained for a "benchmark" circuit used in [16, Example 1] with 3 different values of the resistor

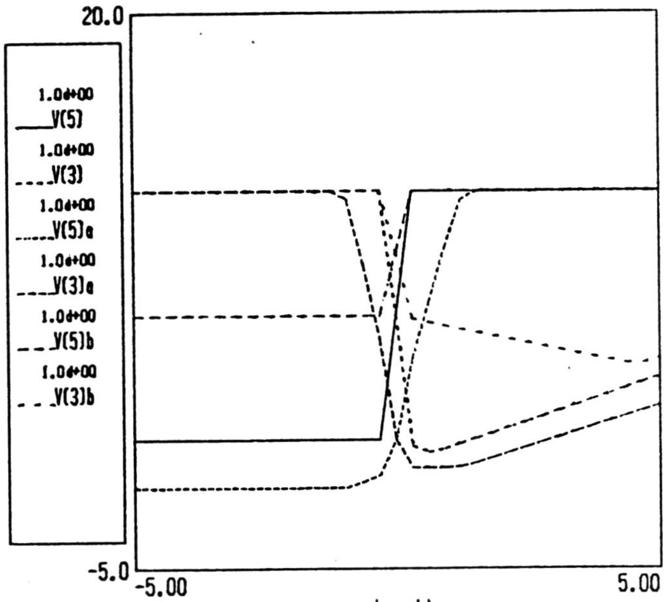

Fig.1. Differential amplifier - DC transfer curves 1.

RE. Fig. 2 shows the same characteristics "extended" in the region of zero input voltage. It can be observed that the only characteristics which satisfy the requirements correspond to curves V(3) and V(5).

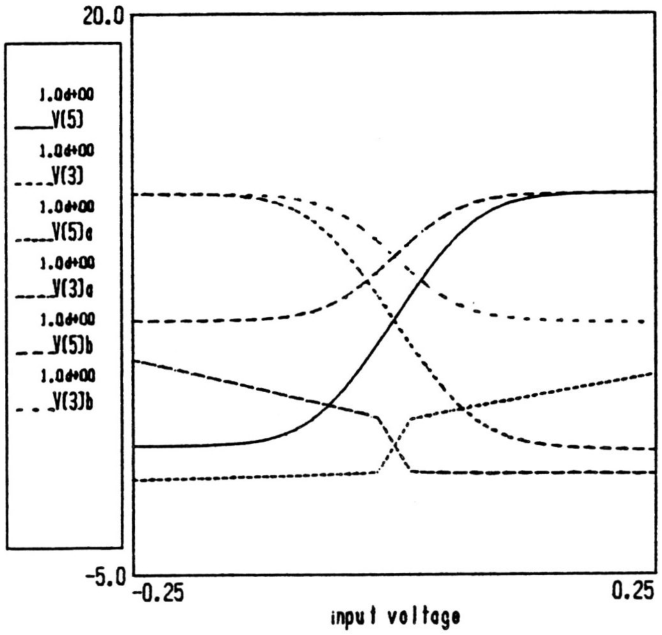

Fig.2. Differential amplifier - DC transfer curves 2.

As an optimization example, a single stage CE amplifier in a self-biasing configuration is analyzed, and it is to find the values of three resistors, R1, R2 and RE [17], such that for the midband frequency f=50KHz, and for the spread of beta.dc between 80 and 250, the voltage gain is as close as possible to the value 10V/V, and the input resistance is not less than 20Kohms. The minimax optimization package WMBG2 [18] used in this example is a modified version of linearly constrained minimization technique combined with routines for numerical approximation of gradients. Fig.3 shows traces of consecutive steps of optimization (without any "external" intervention in this case) in which voltage gain is given as a function of the beta.dc parameter (the influence of input resistance is not shown), and "step 0" denotes the starting characteristic. Also, a "tolerance region" [9.5,10.5] is indicated by its boundaries. It can be observed that after 4 iteration steps the whole characteristic is within the tolerance region, and remaining steps are used for centering the design (one of the centering steps fails but is corrected successfully).

Similar traces for an optimization example are shown in Fig.4 which illustrates an optimization of the same single stage CE amplifier but now the voltage gain

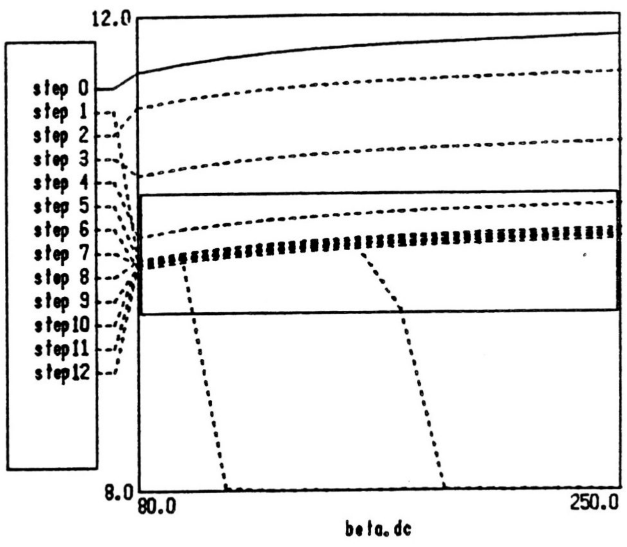

Fig.3. Amplifier optimization - beta.dc variation.

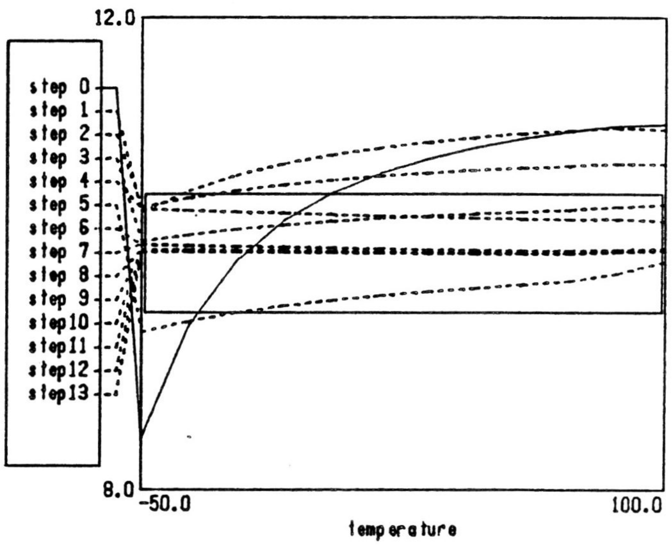

Fig.4. Amplifier optimization - temperature variation.

is stabilized against variation of the temperature (for a fixed value of beta.dc), and it is to find the values of R1, R2 and RE such that for the same midband frequency and for the temperature range [-50,100] degrees Celsius, the voltage gain is as close as possible to 10V/V, and the input resistance is not less than 10Kohm. Again, "step 0" shows the initial design for which the voltage gain variation in the temperature range [-50,100] is approximately 2.5V/V. After 3 iteration steps, the variation is reduced to approximately 0.5V/V, and after 13 iteration steps it is less than 0.1V/V.

6. CONCLUSIONS

An interface "ststandard" graphics systems has been designed for use with the SPICE-PAC package of circuit simulation subroutines. This interfacing package allows the designer to see, in a familiar graphical form, the results of the most important analyses available with the SPICE 2G.6 circuit simulation program, and to interact with the circuit simulation package, dynamically alter circuit variables and parameters, perform analyses, etc.

Although other programs are available which plot the output from the SPICE program, or other circuit analysis programs [3], this package is the only system which allows the full interactive display of SPICE analyses and which permits user interaction to alter the values of circuit elements. Plots from several analyses can be displayed simultaneously. Plots can be scaled and rescaled in several different ways, and the results of the SPICE-PAC analyses, together with the values of circuit variables and other parameters, can be stored and redisplayed.

Future improvements in GRAPH-PAC will allow more extensive interaction with general purpose optimization routines used in conjunction with SPICE-PAC for circuit optimization.

REFERENCES

[1] "DI-3000 User's Guide"; Precision Visuals, Boulder CO, 1984.

[2] "Status Report on the Graphics Standard Planning Committee"; Computer Graphics, vol.13, no.2, 1979.

[3] "VLSI Design Tools - Reference Manual - Release 2.1"; UW/NW VLSI Consortium, University of Washington, Seattle WA 98195, 1984.

[4] D. Bergeron, P. Bono, J.D. Foley, "Graphics programming using the Core system"; ACM Computing Surveys, vol.10, no.4, pp.389-443, 1978.

[5] P.R. Bono, J.L. Encarnacao, R.A. Hopgood, P.J.W. ten Hagen, "GKS - the first graphics standard"; IEEE Computer Graphics and Applications, vol.2, no.5, pp.9-24, 1982.

[6] R.K. Brayton, G.D. Hachtel, A.L. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design"; Proc. of the IEEE, vol.69, no.10, pp.1334-1362, 1981.

[7] G. Chappel, P. Bone, "Core system implementations - a status report"; Computer Graphics, vol.12, no.4, pp.53-65, 1978.

[8] E. Cohen, "Program reference for SPICE 2"; University of California, Berkeley CA 94720, Memo ERL-M592, 1976.

[9] P.E. Gill, W. Murray, S.M. Picken, "The design and structure of a Fortran library for optimization"; ACM Tr. Mathematical Software, vol.5, no.3, pp.259-283, 1979.

[10] A.R. Newton, D.O. Pederson, A.L. Sangiovanni-Vincentelli, C.H. Sequin, "Design aids for VLSI - The Berkeley perspective"; IEEE Tr. Circuit and Systems, vol.28, no.7, pp.666-680, 1981.

[11] D.O. Pederson, "A historical review of circuit simulation"; IEEE Tr. on Circuits and Systems, vol.31, no.1, pp.103-111, 1984.

[12] R.W. Simons, "Minimal GKS"; Computer Graphics, vol.17, no.3, pp.183-189, 1983.

[13] E.L. Sonderegger, "The case for CORE system standardization"; Computer Graphics World, vol.7, no.2, pp.26-30, 1984.

[14] A. Vladimirescu, K. Zhang, A.R. Newton, D.O. Pederson, A. Sangiovanni-Vincentelli, "SPICE Version 2G - User's Guide (10 Aug. 1981)"; Department of Electrical Engineering and Computer Sciences, University of California, Berkeley CA 94720, 1981.

[15] T. Wright, "GKS versus CORE"; Computer Graphics World, vol.7, no.2, pp.18-24, 1984.

[16] W.M. Zuberek, "SPICE-PAC 2G6a.84.05 - User's Guide"; Department of Computer Science, Memorial University of Newfoundland, St. John's, Nfld., Canada A1C 5S7, Technical Report 8404, 1984.

[17] W.M.Zuberek, "SPICE-PAC, a package of subroutines for interactive simulation and optimization of circuits"; Proc. International Conference on Computer Design, Port Chester NY, 1984.

[18] W.M. Zuberek, "WMBG2, a package of subroutines for bounded minimax optimization without derivatives"; Department of Computer Science, Memorial University of Newfoundland, St. John's, Nfld., Canada A1C 5S7, Technical Report 8502, 1985.