1987 Canadian Conference on Very Large Scale Intergration; Winnipeg, Manitoba, October 25–27, 1987, pp.117-120.

PARAMETERIZED SUBCIRCUITS IN THE SPICE-PAC PACKAGE OF SIMULATION SUBROUTINES

W.M. Zuberek

Department of Computer Science, Memorial University
St. John's, NL, Canada A1C–5S7

Abstract

SPICE-PAC is a package of simulation subroutines which is upward compatible with the popular SPICE-2G circuit simulator, i.e., SPICE-PAC accepts SPICE data files and performs the same analyses, but also provides a number of extensions, for example, it supports circuit variables, contains an interface to libraries of standard designs, and allows to enhance the package capabilities by user defined procedures. Parameterized subcircuits generalize the original concept of subcircuits; each parameterized invocation contains a list of substitutions performed during subcircuit expansion. The original requirement of strictly identical subcircuits is thus relaxed to subcircuits with identical topology.

1. INTRODUCTION

Computer-aided circuit analysis or circuit simulation has become a widely accepted tool in the area of integrated circuit design. Using this method circuit designers can easily explore and compare the effects of different designs [1,4]. The SPICE-2 program developed at the University of California at Berkeley [6,7], has become one of the most popular second-generation circuit simulators.

Second generation circuit simulators are usually batch-oriented programs with "closed" sets of operations, and with static definitions of analyses and parameters [2,9]. This is too restrictive in applications that require repeated circuit simulations, for example interactive simulation or circuit optimization [3,4,10]. In such cases a more flexible structure of the circuit simulator is needed, in which different analyses (for the same circuit) can be performed on demand, and which provides an access to internal representation of circuit elements and parameters in order to modify their values. The simulator should thus have a "cloose" structure of a set (or a package) of subroutines rather than a program with one, fixed sequence of operations.

SPICE-PAC 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 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) a hierarchical naming scheme for (nested) subcircuits; subcircuit elements and outputs are denoted by unique qualified names where qualifiers are the path-names of consecutive subcircuits starting from the main circuit level,

(d) access to circuit variables as required in interactive simulation and circuit optimization (there are two types of circuit variables, static and dynamic ones; static variables must be declared within circuit description in order to avoid repeated translations during subsequent analyses; dynamic variables do not require declarations since all references are translated at run time; circuit optimization is a typical application of static variables, while more flexible dynamic variables are required in interactive simulations),

(e) dynamic declarations of parameters and outputs for all analyses,

(f) an interface to libraries of standard modules; standard modules are in the form of subcircuits stored in individual files within a file system.

SPICE-PAC contains 26 interfacing subroutines which control the main operations of the package (and more than 200 internal subroutines and functions which implement these operations). A simple call (or invocation) of an interfacing subroutine (with appropriate parameters) can thus read a circuit description, perform an analysis, modify the circuit, repeat the analysis, etc., as required by a particular application.

Users can also supply their own versions of selected subroutines which replace the corresponding standard SPICE routines, and enhance the original SPICE facilities [11].

One of recent extensions of the package is parameterization of (hierarchical) subcircuits.

2. PARAMETERIZED SUBCIRCUITS

SPICE-like programs support hierarchical subcircuits which means that subcircuit definitions may contain invocations of (other) subcircuits, however, all subcircuit invocations correspond to exact replications of indicated definitions (with appropriate substitutions of node numbers). This may be convenient for digital circuits when the basic "cells" are replicated without any modification (e.g., registers or memories), but in many applications which include analog circuits, even the same subcircuits used in different "environments" usually require adjustments of some element values. In such cases the strict mechanism of SPICE subcircuits is unsatisfactory since it requires repeated definitions for almost identical subcircuits. Parameterization of subcircuits is a generalization of the subcircuit concept that provides the required flexibility (subcircuits without and with parameters may be compared to procedures without and with parameters; if the side effects are not allowed, different invocations of procedures without parameters produce identical results, while invocations of parameterized procedures usually produce results that are different for different invocations).

Another motivation for parameterized (hierarchical) subcircuits is due to decompilation facilities recently added to SPICE-PAC [12] (circuit decompilation converts internal representation of a circuit into its symbolic or source form). Decompilation of circuits modified during interactive simulations or refined by an optimization program, requires a flexible parameterization mechanism which preserves the original (subcircuit) structure of circuits, but which allows to indicate changes of element attributes at any level of (expanded) structures. Parameterized subcircuits are quite satisfactory for this purpose.

Also, a flexible mechanism of subcircuits may be very helpful in accelerating circuit simulation when users indicate the boundaries of partitioning of large circuits for parallel simulation on multiprocessor or distributed systems [8].

The syntax of subcircuit invocations in the SPICE program ("X-type" pseudoelements [9]) is as follows:

```
Xname  nodelist  subcktname
```

In the SPICE-PAC package, parameterized subcircuit invocations may also contain lists of substitutions:

```
Xname  nodelist  subcktname  substitution_list
```

where the "substitution_list" is a sequence of "substitutions" separated by commas, and each "substitution" is in the form "attribute=value". Attribute names are either names of simple subcircuit elements (for those elements which have one attribute only; usually it is the "value" of the element, e.g., the resistance of a resistor), or extended names which are used for multi-attribute elements to indicate polynomial coefficients of nonlinear capacitors and inductors, polynomial coefficients of dependent voltage and current sources, DC, AC and time dependent parameters of independent voltage and current sources, parameters of semiconductor devices, parameters of device models, and also (qualified) parameters of (nested) subcircuit invocations.

During processing of the symbolic circuit description, the input module of SPICE-like programs organizes all circuit elements of the same class (resistors, capacitors, etc.) into list structures composed of element descriptors. The descriptors store all attributes associated with a single circuit element (numerical, e.g., the resistance of a resistor, and textual, e.g., the name of an element). They also contain internal pointers to related data stored in other descriptors, auxiliary tables, etc. Headers of list structures for all classes of circuit elements are organized into a vector of pointers (CEList in Fig.1) which is stored in one of global memory areas, i.e., areas available to many routines and modules of SPICE-PAC. The list of subcircuit invocations corresponds to the 19-th element of CEList, and the list of all subcircuit definitions to its 20-th element (Fig.1).

Fig.1 shows that each invocation descriptor (in the CEList[19] structure) indicates (possibly empty) tables of original and internal node numbers, and a two-column table of "attribute,value" pairs describing the invocation substitutions. Each invocation descriptor also indicates a descriptor in the CEList[20] structure which corresponds to the invoked subcircuit. Each subcircuit definition descriptor contains a pointer to a table of input/output node numbers, and a pointer to an element list which constitutes the subcircuit "body".

It should be noticed that nested invocations (i.e., invocations in subcircuit definitions) initially are not inserted into CEList[19] structure (e.g., XX descriptor in Fig.1); they are appended to this list during subcircuit expansion. This means that after subcircuit expansion there may be several circuit elements with identical (local) element names. In fact, all elements generated during subcircuit expansion are associated with their original definitions and therefore they can be uniquely identified by qualified names which are sequences of consecutive invocation names (or X-names).

3. PARAMETER SUBSTITUTION

The substitutions indicated in parameterized invocations are performed during expansion of subcircuits. The expansion is performed top-down, i.e., it begins at the main circuit level (the CEList[19] from Fig.1 initially contains only main level invocations) and continues through consecutive levels of subcircuit invocations which, during subcircuit expansion, are appended to the CEList[19]:

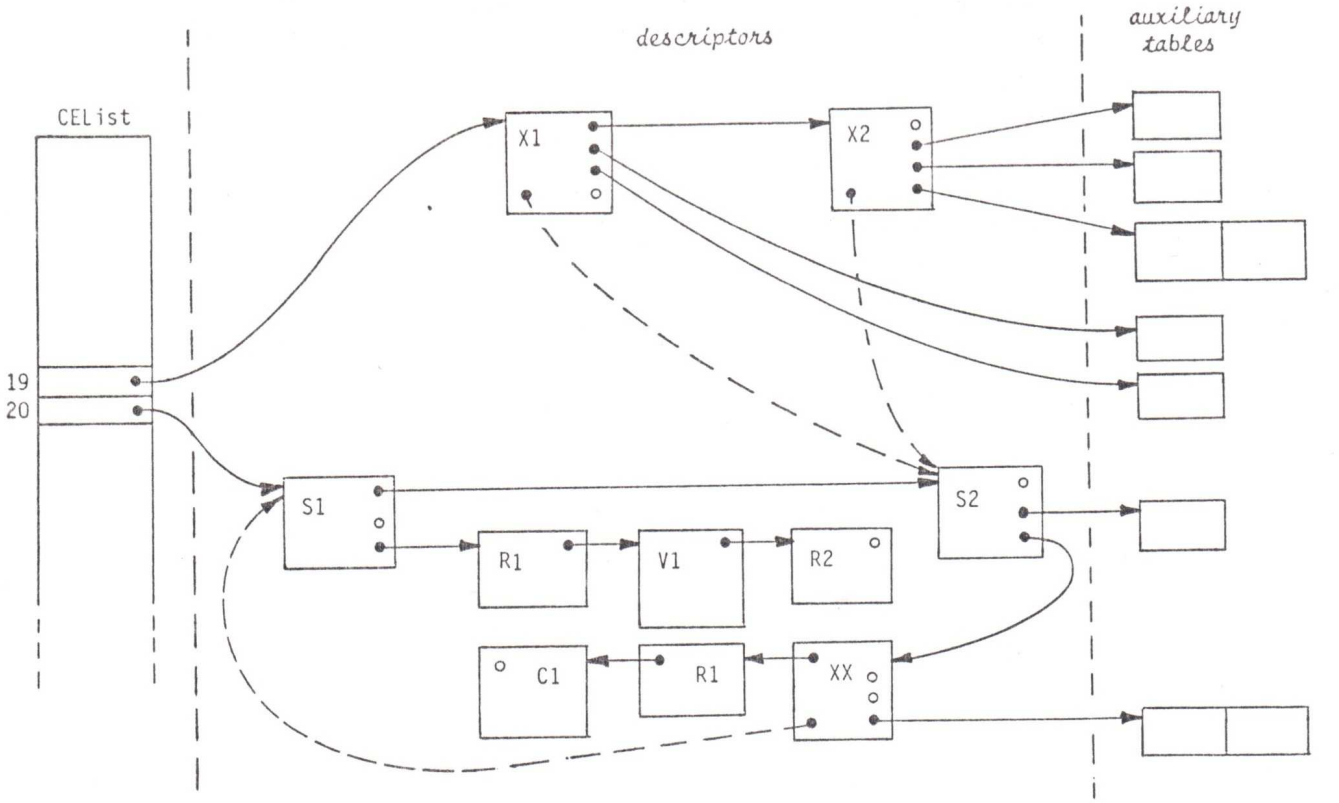

Fig.1. List representation of subcircuit invocations and subcircuit definitions.

```
pointer1 := CEList[19];
while pointer1 ≠ nil do
begin
    pointer2 := definition list of subcircuit descriptor
        indicated by pointer1;
    while pointer2 ≠ nil do
    begin
        set pointer3 to a copy of the element
            indicated by pointer2 and append the
            new desciptor to the corresponding list
            of circuit elements;
        perform node mapping;
        perform parameter substitutions;
        move pointer2 to the next element
    end;
    move pointer1 to the next invocation
end;
```

It should be noticed that the iteration process must be dynamic since the list of subcircuit invocations may extend during processing. Moreover, parameter substitutions "perform parameter substitutions" section) which refer to nested invocations migrate to subsequent invocations and are merged with the corresponding invocation parameters after removing the first (or top) qualifiers. Parameter substitution performed within the descriptor indicated by pointer3 (and identified by "element_name") is controlled by the parameter table indicated by "pointer1":

```
name := element_name indicated by pointer3;
n := length(parameter_table indicated by pointer1);
for i:=1 to n do
if head(parameter_table[i,attribute]) = name then
    if name in X-class then
    begin
        m := length(parameter_table indicated by pointer3);
        ident := tail(parameter_table[i,attribute])
        new := true;
        j := 1;
        while new and j < m do
        begin
            if ident = parameter_table[j,attribute] then
            begin
                parameter_table[j,value] :=
                    parameter_table[i,value];
                new := false
            end;
            j := j+1
        end;
        if new then append (parameter_table[i,attribute];
            parameter_table[i,value]) to parameter_table
            indicated by pointer3
        else store parameter_table[i,value] as indicated
            by parameter_table[i,attribute];
    end;
```

where "head" and "tail" operations applied to a sequence (of qualified names) return the first element and the remaining part of the sequence, respectively.

4. EXAMPLE

As an example, a two-stage amplifier with a second-collector to first-emitter feedback [5,p.433] is shown as a circuit composed of parameterized subcircuits:

```
** subcircuit definition
.SUBCKT STAGE 1,2,3,4
C1 1,5 5U
R3 4,5 150K
R4 5,0 47K
RC 4,2 10K
RE 6,3 4.7K
C3 6,3 50U
Q 2,5,6 QMOD
.MODEL QMOD NPN(BF=50,RB=100)
.ENDS
** main circuit
VS 1,0 AC(1)
R1 4,0 100
R2 6,4 4.7K
C6 3,6 5U
C5 3,10 10U
RL 10,0 25K
X1 1,2,4,5 STAGE
X2 2,3,0,5 STAGE R3=47K,R4=33K,RC=4.7K
VV 5,0 25V
* analyses parameters are irrelevant here
.END
```

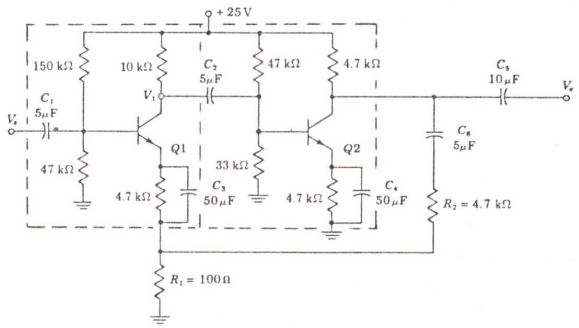

Fig.2. A two-stage amplifier.

5. CONCLUDING REMARKS

The paper describes modification of the original SPICE representation of circuits required for implementation of parameterized subcircuits, and discusses the migration of parameters through consecutive levels of nested subcircuits during subcircuit expansion. It should be noticed that the description is simplified and there are many other details and technicalities which have been neglected since they do not influence the general idea presented in this paper.

Parameterization described in this paper is implemented in the SPICE-PAC package version 2G6c and beyond.

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Operating Grant A8222.

References

[1] P.E. Allen, D.R. Holberg, **CMOS analog circuit design**; Holt, Reinehart and Winston 1987.

[2] E. Cohen, Program reference for SPICE-2; ERL Memorandum M520, Electronics Research Laboratory, University of California, Berkeley CA, 1976.

[3] J.K. Fidler, C. Nightingale, **Computer aided circuit design**; J. Wiley & Sons 1978.

[4] W. Fichtner, M. Morf, **VLSI CAD tools and applications**; Kluwer Academic Publishers 1987.

[5] J. Millman, **Microelectronics - digital and analog circuits and systems**; McGraw-Hill 1979.

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

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

[8] L.T. Smith, D.A. Gross, Preparing large networks for a simulation accelerator; Proc. Int. Symp. on Circuits and Systems, Kyoto, Japan, pp.233-236, 1985.

[9] A. Vladimirescu, K. Zhang, A.R. Newton, D.O. Pederson, A.L. 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.

[10] W.M. Zuberek, SPICE-PAC, a package of subroutines for interactive simulation and optimization of circuits; Proc. IEEE Int. Conf. on Computer Design, Port Chester NY, pp.492-496, 1984.

[11] W.M. Zuberek, P. Gillard, Enhanced circuit simulation using the SPICE-PAC package; Proc. 28 Midwest Symp. on Circuit and Systems, Louisville KY, pp.182-185, 1985.

[12] W.M. Zuberek, Circuit decompilation in the SPICE-PAC package of simulation subroutines; Proc. COMPEURO'87 Conf. on VLSI and Computers, Hamburg, West Germany, pp.229-232, 1987.