View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Memorial University Research Repository

Copyright © 1991 IEEE (DOI 10.1109/MWSCAS.1991.252174).

TABLE–DRIVEN ANALYSES IN THE SPICE–PAC CIRCUIT SIMULATION PACKAGE

W.M. Zuberek and M.S. Zuberek

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

A b s t r a c t

Recent applications of SPICE–PAC to transistor parameter extraction and hierarchical simulation require rather flexible circuit analyses, performed for irregularly distributed values of independent variables. An implementation of table–driven analyses (DC, AC and time–domain) is described in which tables of arbitrarily distributed independent variables (voltages, frequencies or timepoints) are used rather than the fixed–step strategy implemented in SPICE–like simulators. Simple examples are used to illustrate applications of table-driven analyses.

1. INTRODUCTION

SPICE–PAC is a package of circuit simulation routines derived from the popular SPICE–2G circuit simulator from the University of California at Berkeley. It is upward compatible with SPICE, i.e., it accepts the same circuit description language and provides the same set of analyses, but it also contains a number of extensions and enhancements that are not available in SPICE; a hierarchical naming scheme, static and dynamic circuit variables, parameterized subcircuit expansions and elements of higher–level circuit specifications are a few examples of such enhancements. The most distinctive feature of SPICE–PAC is, however, its "open" structure, which means that all analyses as well as (dynamic) definitions of parameters and variables are performed "on demand", as required by specific applications. This also means that SPICE–PAC can easily be integrated with other CAD tools to provide circuit optimization, parameter extraction, statistical simulation, mixed–mode simulation, etc..

Recent applications of SPICE–PAC to transistor parameter extraction and hierarchical simulation (with table–driven specifications of functional blocks) require rather flexible circuit analyses, performed for irregularly distributed values of independent variables. Since SPICE–like simulators support fixed–step analyses only, the required flexibility could not be obtained without significant degradation of the simulation performance. Therefore the implementation of the three fundamental analyses, i.e., DC transfer curve analysis, frequency–domain (AC) analysis and time–domain (TR) analysis, were modified in order to allow specification of domains of independent variables by tables of arbitrarily distributed values rather than a fixed–step strategy. In parameter extraction, such table–driven analyses must be performed at values (voltages, frequencies or timepoints) determined by measurement data; in table–driven specifications of functional blocks, variable spacing of data points is needed for accurate but efficient representation of nonlinearities.

The paper describes the extensions to the input language required for specification of table–driven analyses, and modifications of low–level interfaces for direct control of simulation in integrated applications; it also discusses changes in general organization of basic analyses. Simple examples of table–driven circuit simulation are used as illustrations.

2. INPUT LANGUAGE EXTENSIONS

For DC transfer curve, time–domain and frequency–domain analyses, table-driven options are described by appropriate extensions of the `.DC`, `.TRAN` and `.AC` directives, respectively. The original syntax of these directives is [Vlad]:

```
.DC source_name start_val stop_val increment
.TRAN time_step stop_val [start_val [max_step]] [UIC]
.AC type nrpoints start_val stop_val
```

where lower–case phrases denote user–supplied names and/or values, upper–case fragments must be used verbatim (e.g., TRAN, UIC), and optional parts are enclosed in square brackets "[" and "]".

The table–driven specifications replace the "start", "stop" and "increment" values by a list of values, so the additional formats of these directives are:

```
.DC source_name LIST(list_of_values)
.TRAN LIST(list_of_values) [max_step] [UIC]
.AC list_of_values
```

where `list_of_values` is a sequence of numbers (values of the independent voltage or current source indicated by `source_name` for the DC analysis, timepoints for TR analysis, or frequencies for AC analysis) separated by commas ",". For the time–domain analysis the values must be strictly increasing. For DC analysis the values can be specified in any order, however, preserving an increasing

or decreasing order oif values improves efficiency of analysis, and may also improve convergence properties. For AC analysis, the ordering of values is insignificant.

The following simple example illustrates table–driven analyses:

```
VIN 1 0 AC(1) PWL(0.0 0.0,0.1 1.0,5.0 1.0)
R1 1 2 1.0
C2 2 0 1.0
R2 2 0 1.0
.DC VIN,LIST(0.0,0.2,0.5,1.0)
.PRINT DC V(2)
.TR LIST(0.0,0.1,0.2,0.3,0.5,0.7,1.0,2.0) 0.1
.PRINT TR V(1) V(2)
.AC 0.1,0.2,0.5,1,10,1K
.PRINT AC V(2)
.END

***** DC TRANSFER CURVE

       VIN          V(2)
    0.000D+00    0.000D+00
    2.000D-01    1.000D-01
    5.000D-01    2.500D-01
    1.000D+00    5.000D-01

***** TRANSIENT ANALYSIS

       TIME         V(1)       V(2)
    0.000D+00    0.000D+00  0.000D+00
    1.000D-01    1.000D+00  4.670D-02
    2.000D-01    1.000D+00  1.280D-01
    3.000D-01    1.000D+00  1.950D-01
    5.000D-01    1.000D+00  2.958D-01
    7.000D-01    1.000D+00  3.633D-01
    1.000D+00    1.000D+00  4.251D-01
    2.000D+00    1.000D+00  4.900D-01

***** AC ANALYSIS

       FREQ         V(2)
    1.000D-01    4.770D-01
    2.000D-01    4.234D-01
    5.000D-01    2.685D-01
    1.000D+00    1.517D-01
    1.000D+01    1.591D-02
    1.000D+03    1.592D-04
```

Table–driven analyses can also be indicated in the extended circuit description [Z1]; in this case the lists of values are enclosed in parentheses, as shown in the following example:

```
.END/EXT
.PAR/10 DCTC(VIN,(0.0,0.1,0.5,0.9,1.0))
.PAR/20 TR((0.0,0.1,0.2,0.3,0.5,0.7,1.0,1.5,2.0),0.1,0)
.PAR/30 AC((0.1,0.2,0.5,1.0,100.0))
.END
```

3. INTERNAL INTERFACES

SPICE–PAC is an "open" simulation tool that performs simulation operations "on demand", as required by a specific application [Z1]. SPICE–PAC's operations (such as "define parameters for an analysis", "execute an analysis", etc.) are performed by invocations of corresponding interfacing routines with appropriate parameters [Z2]. Table–driven analyses are indicated by modified invocations of the SPICED, SPICET and SPICEF routines.

SPICED – define parameters for DC analysis

The invocation of SPICED must be equivalent to the following Fortran statement [Z2]:

```
CALL SPICED (source,vistrt,vistop,nrstep,iflag
```

For table–driven DC analysis, `vistrt` is a DOUBLE PRECISION array that contains the list of values of an independent voltage or current source indicated by `source`. The `vistop` argument is immaterial in this case (although it must be DOUBLE PRECISION as well). `nrstep` must be set to the negative number of values in `VISTRT`, and `iflg` is the return flag, as in the original invocation.

SPICET – define parameters for TR analysis

The invocation of SPICET must be equivalent to the following Fortran statement [Z2]:

```
CALL SPICET (tmstrt,tmstop,nrstep,stpmax,incond,iflg)
```

For table–driven TR analysis, `tmstrt` is a DOUBLE PRECISION array that contains the list of required timepoints. The `tmstop` argument is immaterial in this case (although it must be DOUBLE PRECISION). `nrstep` must be set to the negative number of values in `tmstrt`, and the remaining arguments are the same as in the original invocation.

SPICEF – define parameters for AC analysis

The invocation of SPICEF must be equivalent to the following Fortran statement [Z2]:

```
CALL SPICEF (ittab,frtab,nrstep,iflg)
```

For table–driven AC analysis, `frtab` is a DOUBLE PRECISION array that contains the list of required frequencies. The `ittab` argument is immaterial in this case (although it must be INTEGER). `nrstep` must be set to the negative number of values in `frtab`, and `iflg` is the return flag, as in the original invocation.

4. IMPLEMENTATION

DC transfer curve analysis

The DC transfer curve analysis is basically a repetitive DC operating point solution performed for a range of values of one independent voltage or current source in the circuit; the only difference is in starting point which – for each new sweep point of DC analysis – is determined by a linear extrapolation from the previous points.

The general outline of this analysis is as follows [Coh] (`VISRTR`, `VISTOP`, `VIINCR` and `SOURCE` are DC analysis parameters, either defined by the `.DC` directive in the circuit description, or by an invocation of the SPICED routine):

```
initialize;
time:=0;
update_time_dependent_functions_of_independent_
   sources(time);
value:=VISTRT;
count:=0;
while count<NRSTEP do
   update(SOURCE,value);
   solve_the_system_of_circuit_equations;
   if not converged then stop analysis end if;
   store_results;
   value:=value+VIINCR;
   count:=count+1
end while;
```

The modified version uses a logical variable `DCLIST` which is TRUE if the analysis is table–driven, otherwise it is FALSE; `VITAB` is an array that stores the list of values for a table–driven analysis:

```
initialize;
if not DCLIST then value:=VISTRT end if;
time:=0;
update_time_dependent_functions_of_independent_
   sources(time);
count:=0;
while count<abs(NRSTEP) do
   if DCLIST then value:=VITAB[count] end if;
   update(SOURCE,value);
   solve_the_system_of_circuit_equations;
   if not converged then stop analysis end if;
   store_results;
   if not DCLIST then value:=value+VIINCR end if;
   count:=count+1
end while;
```

Time-domain analysis

The time–domain analysis is controlled by a variable timestep mechanism, so the actual timepoints selected for analysis are quite different than the "output timepoints" described in the parameters of this analysis. The (intermediate) solutions, obtained for internal timepoints, are simply stored in a temporary workspace, and in the final stage of analysis are used for linear interpolation of output timepoints. I order to use available workspace efficiently and speed up the analysis, intermediate results are stored for those timepoints only which are needed for final interpolation, i.e., those internal timepoints that "bracket" the output timepoints:

```
initialize;
time:=0;
update_time_dependent_functions_of_independent_
   sources(time);
find_initial_solution;
delta:=TMSTEP;
TIME:=TMSTRT;
advance:=true;
while time<=TMSTOP do
   if time>=TMSTRT do
      if time>TIME then
         store_results;
         TIME:=TIME+TMSTEP;
         advance:=true
      else if advance then
         store_results;
         advance:=false
      else
         replace_previous_results
      end if
   end if;
   adjust_time_and_iterate_solution_until_converged_
      or_timestep_too_small(time,delta)
end while;
```

The modified version uses a logical variable `TMLIST` which is TRUE if the analysis is table–driven, otherwise it is FALSE, and an array `TMTAB` that stores the list of values for a table–driven analysis:

```
initialize;
time:=0;
update_time_dependent_functions_of_independent_
   sources(time);
find_initial_solution;
delta:=TMSTEP;
if not TMLIST then TIME:=TMSTRT
else
   count:=1;
   TIME:=TMTAB[count]
end if;
advance:=true;
while time<=TMSTOP do
   if time>=TMSTRT do
      if time>TIME then
         store_results;
         if TMLIST then
            count:=count+1;
            TIME:=TMTAB[count]
         else TIME:=TIME+TMSTEP end if
         advance:=true
      else if advance then
         store_results;
         advance:=false
      else
         replace_previous_results
      end if
   end if;
   adjust_time_and_iterate_solution_until_converged_
      or_timestep_too_small(time,delta)
end while;
```

Frequency-domain analysis

The frequency–domain analysis can be performed with either linear or logarithmic variation; the actual choice (the `type` argument of the `.AC` line) is represented by a logical variable `ACVAR` which is TRUE for the linear variation and is FALSE otherwise:

```
initialize;
if ACVAR then delta:=(FRSTOP-FRSTEP)/(NRSTEP-1)
else delta:=exp(ln(FRSTOP/FRSTRT)/(NRSTEP-1)) end if;
freq:=FRSTRT;
count:=0;
while count<NRSTEP do
   solve_the_system_of_linearized_circuit_
      equations(freq);
   store_results;
   if ACVAR then freq:=freq+delta
   else freq:=freq*delta end if;
```

```
   count:=count+1
end while;
```

The modified version uses a logical variable ACLIST which is TRUE if the analysis is table–driven, otherwise it is FALSE; FRTAB is an array that stores the list of values for a table–driven analysis:

```
initialize;
if not ACLIST then
  if ACVAR then delta:=(FRSTOP-FRSTEP)/(NRSTEP-1)
  else delta:=exp(ln(FRSTOP/FRSTRT)/(NRSTEP-1)) end if;
  freq:=FRSTRT
end if;
count:=0;
while count<NRSTEP do
   if ACLIST then freq:=FRTAB[count] end if;
   solve_the_system_of_linearized_circuit_
      equations(freq);
   store_results;
   if not ACLIST then
      if ACVAR then freq:=freq+delta
      else freq:=freq*delta end if
   end if;
   count:=count+1
end while;
```

5. EXAMPLE

The following example illustrates table–driven DC analysis used for characterization of the transfer curve of a MOSFET inverter:

```
*The design and analysis of VLSI circuits (Glasser...)
.OPTIONS DEFL=2.25E-6
VIN 1 0 0
VDD 9 0 5
* MOS subcircuit
M1 2 1 0 0 NENHS W=11.2U AD=61P PD=42U
.MODEL NENHS NMOS LEVEL=3 RSH=0 TOX=330E-10 LD=0.19E-6
+      UO=650 XJ=0.27E-6 VMAX=13E4 ETA=0.25 KAPPA=0.5
+      NSUB=5E14 THETA=0.1 VTO=0.946 CGSO=2.43E-10
+      CGDO=2.43E-10 CJ=6.9E-5 CJSW=3.3E-10 PB=0.7
+      MJ=0.5 MJSW=0.3 NFS=1E10
M2 9 2 2 0 NDEPS W=4.2U L=6.25U
.MODEL NDEPS NMOS LEVEL=3 RSH=0 TOX=330E-10 LD=0.19E-6
+      UO=650 XJ=0.27E-6 VMAX=13E4 ETA=0.25 KAPPA=0.5
+      NSUB=50E14 THETA=0.04 VTO=-2.078 CGSO=2.43E-10
+      CGDO=2.43E-10 CJ=6.9E-5 CJSW=3.3E-10 PB=0.7
+      MJ=0.5 MJSW=0.3 NFS=1E10
.DC VIN,LIST(0.0,0.4,0.8,0.9,1.0,1.1,1.2,1.25,1.3,1.35,
+  1.37,1.38,1.40,1.425,1.45,1.475,1.5,1.55,1.57,1.58,
+  1.6,1.625,1.65,1.7,1.8,2.0,2.2,2.5,3.0,3.75,5.0)
.PRINT DC V(2)
.END
```

Fig.1 shows the transfer curve of this inverter and the points used in table–driven DC analysis.

The results of this analysis are used in a table–driven controlled source that models the behavior of the inverter [ZZ], and compares the "reference" MOSFET results with two table–driven controlled sources modeling the inverter, (i) E1 that uses linear interpolation, and E2 that uses quadratic interpolation:

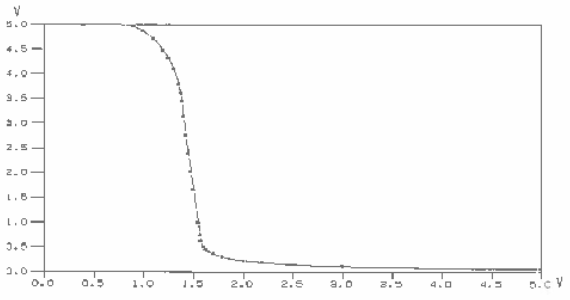

Fig.1. MOS inverter's transfer curve.

```
*MOSFET inverter as a table-driven source
.OPTIONS DEFL=2.25E-6  LIMPTS=501
VIN 1 0 0
VDD 9 0 5V
* MOS subcircuit (reference level)
M1 2 1 0 0 NENHS W=11.2U AD=61P PD=42U
.MODEL NENHS NMOS LEVEL=3 RSH=0 TOX=330E-10 LD=0.19E-6
+      UO=650 XJ=0.27E-6 VMAX=13E4 ETA=0.25 KAPPA=0.5
+      NSUB=5E14 THETA=0.1 VTO=0.946 CGSO=2.43E-10
+      CGDO=2.43E-10 CJ=6.9E-5 CJSW=3.3E-10 PB=0.7
+      MJ=0.5 MJSW=0.3 NFS=1E10
M2 9 2 2 0 NDEPS W=4.2U L=6.25U
.MODEL NDEPS NMOS LEVEL=3 RSH=0 TOX=330E-10 LD=0.19E-6
+      UO=650 XJ=0.27E-6 VMAX=13E4 ETA=0.25 KAPPA=0.5
+      NSUB=50E14 THETA=0.04 VTO=-2.078 CGSO=2.43E-10
+      CGDO=2.43E-10 CJ=6.9E-5 CJSW=3.3E-10 PB=0.7
+      MJ=0.5 MJSW=0.3 NFS=1E10
* table-driven voltage-controlled voltage sources
E1 3 0 PWL(1) 1 0  USE(Tdata)
R1 3 0 1K
E2 4 0 PWQ(1) 1 0  USE(Tdata)
R2 4 0 1K
.TABLE Tdata (0.00 5.0000,0.400 5.0000,0.800 4.9981,
+  0.90 4.9578,1.00 4.8630,1.10 4.7075,1.20 4.4726,
+  1.25 4.3109,1.30 4.1007,1.35 3.7945,1.37 3.5996,
+  1.38 3.4456,1.4 3.1383,1.425 2.7442,1.45 2.3773,
+  1.475 2.0058,1.5 1.6607,1.55 .99524,1.57 .74334,
+  1.58 .61411,1.60 .50614,1.625 .44914,1.65 .41187,
+  1.70 .35969,1.80 .29606,2.00 .22661,2.20 .18704,
+  2.50 .15072,3.00 .11645,3.75 .08945,5.00 .06739)
.DC VIN 0 5 0.01
.PRINT DC V(2) V(3,2) V(4,2)
.END
```

The results of these comparisons, i.e., the differences between the original MOSFET inverter transfer characteristic and the table–driven models, are shown in Fig.2 and Fig.3, for linear and quadratic interpolation, respectively.

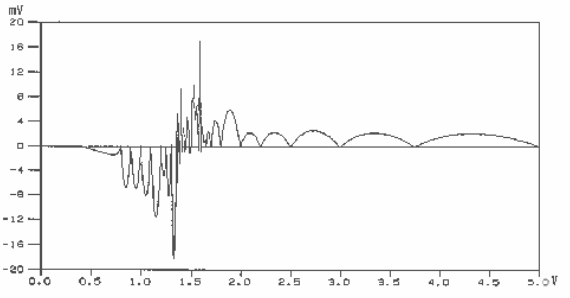

Fig.2. Approximation error for linear interpolation.

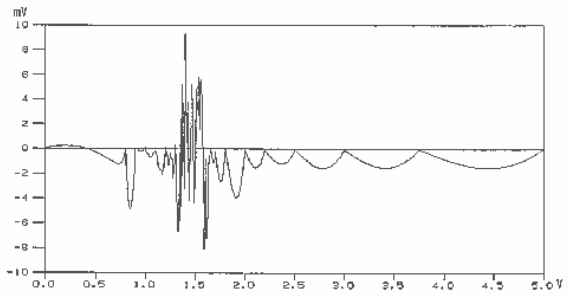

Fig.3. Approximation error for quadratic interpolation.

The approximation errors are at the level of 1 percent or less, and for quadratic interpolation they are approximately two times smaller than for linear interpolation.

6. CONCLUDING REMARKS

Uniformly distributed simulation results, typical for SPICE–like simulators, are inadequate for applications, in which the values of independent variables are arbitrarily distributed. Examples of such applications include simulation–based parameter extraction and higher–level simulation. In parameter extraction, measurement data determine the values of independent variables (voltages, currents, time or frequency) for which the simulation results are needed, so the simulator must be flexible enough to accept arbitrarily distributed measurement data. In higher–level simulation, characteristics of functional blocks are represented by (multidimensional) data–driven elements, in which the accuracy of representation determines the distributions of data points, and the simulators should be flexible enough to deal with such requirements. Table–driven analyses provide simulation flexibility that is required in these applications.

The simple example of MOS inverter modeling shows that data-driven elements can provide accuracy needed in practical applications, at the same time reducing significantly the computational effort required for evaluation of complex models of semiconductor devices [BVS,Rau].

Acknowledgement

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

R e f e r e n c e s

[BVS] J. Barby, J. Vlach, K. Singhal, "Optimized polynomial splines for FET models" Proc. 1984 Int. Symp. on Circuits and Systems, Montreal, Canada, pp.1159-1162.

[Coh] E. Cohen, "Program reference for SPICE 2"; Memorandum UCB/ERL M592, University of California, Berkeley, CA 94720, 1976.

[GD] L.A. Glasser, D.W. Dobberpuhl, "The design and analysis of VLSI circuits"; Addison–Wesley 1985.

[Rau] K-G. Rauch, "A table model for circuit simulation"; Proc. ESSCIRC '86, pp.211–213, 1986.

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

[Z1] W.M. Zuberek, "SPICE–PAC version 2G6c – an overview"; Technical Report #8903, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada A1C–5S7, 1989.

[Z2] W.M. Zuberek, "SPICE-PAC version 2G6c – user's guide"; Technical Report #8902, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada A1C–5S7, 1989.

[ZK] W.M. Zuberek, A. Konczykowska, "FIT–2, an extraction program based on the SPICE–PAC simulation software"; Proc. 34th Midwest Symp. on Circuits and Systems, Monterey CA, pp.142-145, 1991,

[ZZ] M.S. Zuberek, W.M. Zuberek, "Enhanced controlled sources as device models in the SPICE–PAC simulation package"; Proc. 30th Midwest Symp. on Circuits and Systems, Syracuse NY, pp.603–606, North–Holland 1988.