Int. Symp. on Circuits and Systems (ISCAS'89), Portland, OR, May 9–11, 1989, pp.709-712.

DISTRIBUTED TRANSIMISSION LINES AND TIME-DOMAIN ANALYSIS IN SPICE-LIKE CIRCUIT SIMULATORS

W.M. Zuberek[†], A. Konczykowska[‡], H. Wang[‡]

[†] Department of Computer Science
Memorial University
St. John's, NL, Canada A1C-5S7

[‡]Laboratoire de Bagneux
Centre National d'Etude des Telecommunications
196 rue Ravera, 92220 Bagneux, France

Abstract

Several problems arising in the SPICE implementation of the time-domain analysis of circuits with distributed transmission lines are indicated, and it is shown that some simple modifications of the original handling of transmission lines can significantly reduce both memory requirements and CPU time needed for this analysis. The modifications allow to "trade" simulation time for accuracy of results. In effect, results more accurate than the original SPICE ones can be obtained with much smaller computational effort. An example of microwave oscillator is used as an illustration of proposed improvements.

1. INTRODUCTION

Circuit simulation, or computer-aided circuit analysis, is one of those computer-aided design applications that have become well established and widely accepted in the design of electronic circuits. Using circuit simulators, the designers can easily determine the functionality and performance of circuits before the expensive and time-consuming fabrication takes place.

One of the most useful but computationally most complex tasks of circuit simulation is the time-domain analysis of dynamical circuits, i.e., circuits containing capacitors and inductors [4,5,6]. It requires a (numerical) solution of a set of (nonlinear) ordinary differential equations describing the circuit. The detailed algorithm depends upon the integration method used, but - generally - the simulation interval is divided into (usually variable) timesteps (sometimes called internal timesteps), and at each timepoint the solution is obtained using the information from previous timepoints. SPICE-class simulators use iterative "implicit" [1,6] integration methods which provide much "better" stability regions than simpler but not so reliable "explicit" methods. The two most popular "implicit" methods are the trapezoidal rule and variable order backward differentiation (or Gear's) method, the second designed specifically to deal with stiff differential equations [3].

The time-domain analysis in SPICE-like simulation programs is controlled by two mechanisms [1], the variable timestep used to minimize integration errors, and the breakpoint table which contains all those time instances in which "special" changes of voltages and/or currents occur (all boundary points of PWL time-dependent source functions are good examples of "breakpoints").

It appears that the implementation of transmission lines in SPICE-like simulators may result in excessive memory and simulation time requirements for time-domain analysis of even quite simple circuits. This is due to very large numbers of breakpoints that can be generated by transmission lines, and which increase both the memory requirements and simulation time needed for the analysis. A number of improvements are proposed to reduce these requirements, and their effects are compared with the original SPICE simulation of transmission lines. Significant reductions in required memory as well as computational effort are obtained.

The proposed modifications have been implemented in SPICE-PAC [8], a simulation package that is upward compatible with the SPICE-2G6 simulators. This means that SPICE-PAC accepts the same circuit descriptions (with several minor exceptions) and it provides the same set of circuit analyses as SPICE, but it also contains a number of extensions which are not available in the SPICE programs, for example (static and dynamic) circuit variables, parameterized subcircuits, or enhanced circuit elements and analyses. However, the main difference between SPICE and SPICE-PAC is in their structure; SPICE is a "closed" program with a fixed set of circuit elements and circuit analyses while SPICE-PAC is an"open" package that can easily be combined with other CAD tools, for example, optimization methods, symbolic and statistical simulators, circuit extractors and so on.

The paper starts with a brief outline of time-domain analysis, as implemented in SPICE-like simulators. This outline, given in section 2, emphasizes coordination of variable timesteps with the breakpoint table, two basic mechanisms used in this analysis. Section 3 describes the original implementation of transmission lines; it also contains a critique of this implementation. The modified implementation of transmission lines is presented in section 4, while section 5 illustrates these modifications with results obtained for a microwave oscillator. Section 6 contains concluding remarks.

2. AN OUTLINE OF TIME-DOMAIN ANALYSIS

In SPICE-like programs, the time-domain (or "transient") analysis is performed in two consecutive steps, the so called "Initial Transient" analysis and the proper "Transient" analysis [1]. The initial transient analysis finds the "initial" solution (i.e., the solution of circuit equations for the time equal to zero) with default or user-specified initial conditions. The proper transient analysis integrates the differential (nonlinear) circuit equations from one timepoint to another, using either the trapezoidal rule (default) or backward differentiation (or Gear) method.

The proper transient analysis is controlled by the variable timestep as well as the so called "breakpoint table" (BPT). The variable timestep is controlled by iteration count and an estimated truncation error [1]. The iteration count uses the number of Newton-Raphson iterations required to converge at a given timepoint; if this number is less than the parameter (or "OPTION") ITL3, the timestep is doubled provided it does not exceed the value of Tstepmax. Tstepmax is either specified by users (TMAX in [7]), or (by default) is equal to

$$\text{Tstepmax} = (\text{Tstop} - \text{Tstart})/50$$

where "Tstop" and "Tstart" are parameters of transient analysis (TSTOP and TSTART in [7]). Moreover, if the number of Newton-Raphson iterations is greater than the limit ITL4, the iteration is terminated as nonconvergent, the "Timestep" is divided by 8, and the iterative solution begins for a new timepoint (determined by this new Timestep) provided the reduced timestep is greater than the minimum timestep

$$\text{Tstepmin} = 10^{-9} * \text{Tstepmax}$$

(otherwise the message "internal timestep too small" is reported and the analysis terminates [1]).

The coordination of (variable) timesteps with the breakpoints stored in BPT is done within the following framework of time-domain analysis [1] (the following "high-level" code uses "if-then-endif", "for-do-endfor" and "while-do-endwhile" control structures):

```
   Time:=0;
   create_breakpoint_table(BPT);
   Nbpt:=length(BPT);
   set_time_dependent_source_functions(Time);
   Initial_Transient:
   Ibpt:=1; (* the first breakpoint is always zero *)
   Break:=true;
   Timestep:=Tstepmax;
   Laststep:=Timestep;
storeresults:
   store_the_solution;
   if Time > Tstop then
      interpolate_output_results_and_return
   else if Break then
      Ibpt:=Ibpt+1;
      Newstep:=0.1*min(Laststep,BPT[Ibpt]-Time);
      Timestep:=min(Timestep,Newstep);
      if Ibpt = 2 then Timestep:=0.1*Timestep endif;
      Break:=false
   else if Time+Timestep > BPT[Ibpt] then
      Laststep:=Timestep;
      Timestep:=BPT[Ibpt]-Time;
      Break:=true
   endif;
newtimepoint:
   Time:=Time+Timestep;
   set_time_dependent_source_functions(Time);
   solve_the_system_of_circuit_equations;
   if converged then
      Oldstep:=Timestep;
      estimate_integration_error_and_adjust(Timestep);
      if integration_error_is_acceptable then
         go to storeresults endif
      Time:=Time-Oldstep
   else
      Time:=Time-Timestep;
      Timestep:=Timestep/8
   endif;
   if Timestep > Tstepmin then go to newtimepoint endif;
   stop_analysis("timestep too small");
```

"Time" is the simulated time, "BPT" is the breakpoint table of size "Nbpt", "Ibpt" is an index in BPT, "Break" is a logical flag that indicates when a breakpoint (from BPT) has been used, and then the timestep is reduced at least 10 times (and it is reduced once more 10 times for the initial step) in anticipation of some "extra" changes at the breakpoint. Consequently, if there are many breakpoints, the (internal) timestep is kept small not because of integration errors, but because of expected voltage and/or current changes at breakpoints. This obviously increases the number of integration steps and slows down the simulation.

3. ORIGINAL TRANSMISSION LINES

There are two basic aspects of handling transmission lines in circuit simulators; one is to implement the line delay, and the second is to perform circuit analyses in those instants of time in which some voltages and/or currents change. The delay is usually implemented by a local "history" (associated with a transmission line) which stores all line outputs evaluated within the last time interval that is equal to the line delay (it is thus a "temporal window" associated with the element). The actual "delayed" information that is needed in formulation of circuit equations, is obtained by an interpolation of "history" (and after each new solution, the "history" is updated by storing new results and deleting the data that are "older" than the delay).

The second aspect is implemented through the breakpoint table which contains all circuit's "characteristic" time instants. The effect of transmission line delays is introduced in this table by adding new breakpoints which are obtained from existing ones by adding the delay values.

The breakpoint table is thus created in two phases. The first phase analyzes all independent voltage and current sources, and collects the information extracted from the time-dependent source functions:

```
set_empty(BPT);
append(BPT,0);
append(BPT,Tstop);
for each independent_voltage_and_current_source do
   case (time_dependent_function_of_the_source) of
   "PULSE" : time:=0;
             while (time < Tstop) do
                append(BPT,time+start_of_rise);
                append(BPT,time+end_of_rise);
                append(BPT,time+start_of_fall);
                append(BPT,time+end_of_fall);
                time:=time+pulse_period
             endwhile;
   "SINE" : append(BPT,delay_time);
   "EXP" : append(BPT,start_of_rise);
           append(BPT,start_of_fall);
   "PWL" : for each point do
              if (time_coordinate < Tstop) then
                 append(BPT,time_coordinate)
              endif
           endfor
   endcase
endfor;
```

where "append(BPT,x)" increases the size of BPT by one element and stores the value "x" at the end of BPT.

The second phase introduces the contributions of transmission lines:

```
Tol:=0.01*Tstepmax;
for each transmission_line do
   Tdel:=time_delay_of_the_transmission_line;
   Nbpt:=length(BPT);
   for j:=1 to Nbpt do
      time:=BPT[j]+Tdel;
      while time < Tstop do
         append(BPT,time);
         time:=time+Tdel
      endwhile;
      sort(BPT);
      compress(BPT,Tol)
   endfor
endfor;
```

where "sort(BPT)" rearranges the elements of BPT in the ascending order, and "compress(BPT,Tol)" removes all those elements of BPT which differ from the preceding element by less than "Tol" (and contracts the size of BPT accordingly).

It can be observed that this scheme has a number of disadvantages:

- the number of generated breakpoints depends upon ordering of transmission lines in the circuit description; since any "next" line generates new breakpoints starting from all "previous" breakpoints, i.e., also breakpoints introduced by "previous" transmission lines, there is an "asymmetry" of

contributions which can change the number of generated breakpoints (and the results of analysis) when the circuit elements are reordered,

- for circuits with more than one transmission line and with several different line delays which are not trivially dependent (e.g., one delay is twice as large as the other), the number of generated breakpoints grows nonlinearly with the length of the analysis interval, and this nonlinearity increases with the number of transmission lines (with different delays) in the circuit,
- the total number of generated breakpoints can easily become excessive (for the example in section 5, the number of "compressed" breakpoints for the 10 ns simulation interval and for the default Tstepmax, is equal to 20,964 while the maximum number of breakpoints before compression is equal to 255,513; for Tstepmax=Tstep=10 ps the number of "compressed" breakpoints increases to 67,284),
- the large number of generated breakpoints rapidly increases the number of (internal) timepoints and slows down the simulation; in fact, in situations when the number of breakpoints is large, it is more reasonable to perform the time-domain analysis with a reduced (internal) timestep (and no breakpoints) rather than with very many breakpoints that systematically "disturb" the integration process,
- the large number of (internal) timepoints increases the memory requirements since all intermediate solutions are stored for the final interpolation of results.

The last point appears to be the easiest one to deal with. An improvement (proposed some time ago by Durbin and Haussy [2], although in a different context) reduces the workspace required for intermediate solutions. Since the final (linear) interpolation of results needs only two intermediate points which "bracket" each of the (external) equidistant points determined by the "Tstep" parameter, all other solutions are redundant and can be removed from the workspace.

4. MODIFIED TRANSMISSION LINES

In the modified handling of transmission lines, the number of generated breakpoints is radically reduced by disregarding the "secondary" breakpoints, i.e., all those breakpoints that are generated as a result of other breakpoints generated by transmission lines. Consequently, the number of modified contributions to the breakpoint table is linearly dependent upon the length of the analysis interval, and this number does not depend upon the ordering of circuit elements.

Moreover, the total number of breakpoints contributed by transmission lines is limited by a value proportional to the number of "nominal" timesteps (and also the length of the analysis interval). If the number of generated breakpoints exceeds this limit, all contributions are deleted and instead the time-domain analysis is performed with the reduced maximum internal timestep "Tstepmax":

```
Tol:=0.01*Tstepmax;
Nbpt:=length(BPT);
Limit:=P1*(Tstop/Tstepmax);
Count:=0;
for each transmission_line do
   Tdel:=time_delay_of_the_transmission_line;
   for j:=1 to Nbpt do
      time:=BPT[j]+Tdel;
      while time < Tstop do
         append(BPT,time);
         time:=time+Tdel;
         Count:=Count+1;
         if P1 > 0 and Count > Limit then
            contract(BPT,Nbpt);
            Tstepmax:=Tstepmax/P2;
            go to continue
         endif
      endwhile
   endfor
endfor;
continue:
   sort(BPT);
   compress(BPT,Tol)
```

where "contract(BPT,Nbpt)" contracts "BPT" to "Nbpt" elements, i.e., it removes all breakpoints contributed by transmission lines; "P1" and "P2" are (new) parameters defined by OPTIONS (with default values equal to 2 and 5, respectively).

Several new options have been added to the input language (or circuit description language) in order to provide more flexible handling of transmission lines:

NOTLBP - to disregard breakpoints generated by transmission lines,

TLPAR1=p1 - to define the value of parameter "P1" (this definition overrides the NOTLBP option),

NOTLTS - to ignore the reduction of the timestep in the case of too many generated breakpoints,

TLPAR2=p2 - to define the value of parameter "P2" (this definition overrides the NOTLTS option),

TLINES - to return to the original, SPICE handling of transmission lines.

5. EXAMPLE

A Heterojunction Bipolar Transistor (HBT) oscillator is used as an illustration of improvements obtained by the modifications proposed in this paper. It is composed of one HBT, three lumped R, L and C elements, and three transmission lines with delays from 70 to 170 ps, as shown in Fig.1. Transient analysis is performed over time interval of 10 ns with the step of 10 ps:

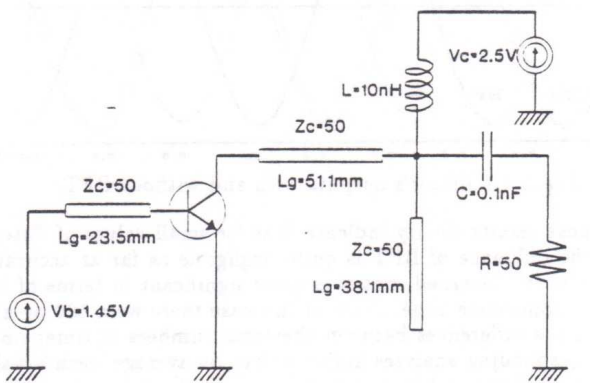

Fig.1. The HBT oscillator.

```
* HBT microwave oscillator -- CNET-Bagneux
.OPT ITL5=0  LIMPTS=1001
.TRAN 10PS 10NS
.PRINT TRAN V(8)
IC 4 0 PULSE(0,0.02,0,0.1N,0.1N,0.1N)
QH 4 2 0 HBT
T1 2 0 1 0 Z0=50 TD=78.3PS
VB 1 0 DC 1.4523
T2 4 0 5 0 Z0=50 TD=170.3PS
T3 5 0 6 0 Z0=50 TD=127PS
L 5 7 10N
VC 7 0 DC 2.5
```

```
C 8 0.1N
R 8 0 50
.MODEL HBT NPN(... a list of parameters ...)
.END
```

The original TRANSIENT analysis (as shown above) performed by a SPICE-like simulator required 281 min of a VAX-11/780 computer (running 4.3 BSD-UNIX) and approximately 4MB of workspace (for creation of the breakpoint table; it needed only 2MB of workspace for the analysis). The breakpoint table contained 20,964 entries, generation of which required almost 10.0 min of CPU time.

In order to indicate the influence of the breakpoint table, two series of analyses were performed for the modified transmission lines; one with (modified) contributions of transmission lines (as described in section 4; with P1=100 to avoid the limitation effects), and the second without these contributions (i.e., with the NOTLBP option). The analyses were performed with Tstep=10 ps, Tstop=10 ns, and Tstepmax set to 10 ps, 5 ps and 2 ps. The results are compared in Fig.2 in which the curves for Tstepmax=5 ps are "shifted" by 0.2 ns, and those for Tstepmax=2 ps are "shifted" by 0.4 ns. It can be observed that for Tstepmax=5 ps the two curves are practically identical (indicating that the breakpoints introduced by transmission lines are insignificant), while for Tstepmax=2 ps the two corresponding curves overlap almost everywhere. The simulation times (in minutes) and the total numbers of evaluated timepoints are summarized in the following table:

	Tstepmax:	10 ps	5 ps	2 ps
with TL contributions	time:	3.2	4.6	9.0
	points:	2,417	3,435	6,624
without TL contributions	time:	1.5	3.0	7.0
	points:	1,016	2,016	5,016

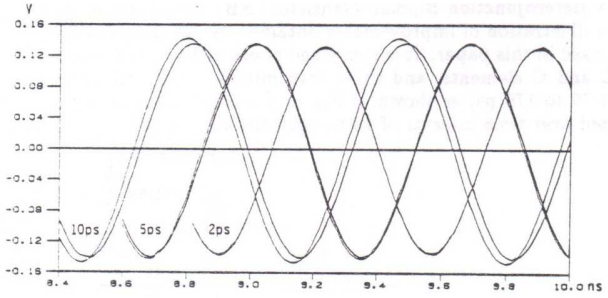

Fig.2. Oscillator's outputs with and without BPT.

These results clearly indicate that for small values of Tstepmax the influence of BPT is quite negligible as far as accuracy of results is concerned, but it is quite significant in terms of increased simulation time. Since in this case there were 506 breakpoints, the differences between the total numbers of timepoints for corresponding analyses indicate that, on average, each breakpoint introduces 3 timepoints (extending the total simulation time correspondingly).

Fig.3 illustrates the influence of the Tstepmax parameter; it shows a family of five curves, the "standard" SPICE results denoted by "STD", and results of (modified) time-domain analysis (without generation of TL breakpoints) for Tstepmax equal to 5 ps, 2 ps, 1 ps, and 0.5 ps, obtained in 3.0 min, 6.3 min, 13.6 min and 32.0 min, respectively. It can be observed that the results for Tstepmax=0.5 ps are very close to the STD ones, but they are obtained almost seven times faster. For less accurate results the "speed up" can be even two orders of magnitude.

All results for modified transmission lines were obtained with 100KB of workspace.

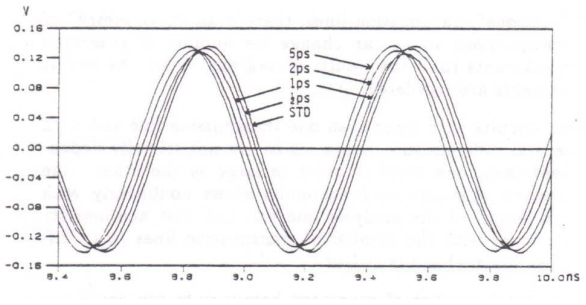

Fig.3. Original and modified oscillator's outputs.

6. CONCLUDING REMARKS

It has been shown that the original SPICE implementation of the time-domain analysis of circuits with transmission lines can result in excessive memory and simulation time requirements for even very simple circuits. Such requirements are due to very large numbers of breakpoints which are generated when the analyzed circuit contains several transmission lines with different delays, and the delays are much smaller than the analysis interval.

These excessive requirements can be substantially reduced by relatively straightforward modifications of contributions which transmission lines introduce in the breakpoint table. Furthermore, due to modified handling of transmission lines, approximate results of time-domain analysis can be obtained with enormous reductions of the simulation time. More accurate results are available at the expense of simulation time with almost linear relationship between the simulation time and the maximum (internal) timestep.

The modifications presented in this paper have been implemented in the SPICE-PAC version 2G6c:89.01 and beyond.

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Operating Grant A8222, and Northern Telecom through Memorial University Interaction Program.

References

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

[2] F. Durbin, J. Haussy: "SPICE-PAC and SPICE-PAC PLUS - corrections and improvements"; Internal Documentation (unpublished).

[3] C.W. Gear, "Simultaneous numerical solution of differential-algebraic equations"; IEEE Trans. on Circuit Theory, vol.18, no.1, pp.89-95, 1971.

[4] W.J. McCalla, "Fundamentals of computer-aided circuit simulation"; Kluwer Academic Publ. 1988.

[5] D.O. Pederson, "Computer aids in integrated circuit design"; in: "Computer Design Aids for VLSI Circuits", P. Antognetti, D.O. Pederson, H. de Man (eds), Sijthoff and Noordhoff 1981.

[6] J. Vlach, K. Singhal, "Computer methods for circuit analysis and design"; Van Nostrand Reinhold 1983.

[7] 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.

[8] W.M. Zuberek, "SPICE-PAC, a package of subroutines for interactive simulation and circuit optimization"; Proc. IEEE Int. Conf. on Computer Design (ICCD-84), Port Chester, NY, pp.492-496, 1984.