23-rd Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 30 Oct.–1 Nov. 1989, pp.445-449.

Mixed–Mode Simulation with SPICE-like Circuit Simulators

W.M. Zuberek

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

Abstract

An extension to the popular SPICE circuit simulator is described that analyses analog circuits with digital components represented at the gate, functional or behavioral levels. The analog-to-digital conversion is performed by (multilevel) "thresholders" which assign digital signals to subranges of voltages determined by a set of voltage thresholds. The digital-to-analog conversion is implemented by voltage sources with "controlled" characteristics that approximate (discrete) digital signals by continuous piecewise linear functions. The paper outlines internal organization of mixed, analog/digital simulation, presents modification of analyses and extensions to the input language needed for analog-digital interactions, and describes the specification of digital circuits.

1. INTRODUCTION

The phrase "mixed-mode" or "mixed analog/digital" simulation has been used to refer to the simulation of electrical networks consisting of both analog and digital parts, regardless of the level of design abstraction. Design abstraction, in this context, means the level at which the network is specified for simulation; it can be device, circuit, functional or behavioral level [MAR].

The popularity of mixed-mode simulation [CoWi,MAR,SaDi,Samp] is due to analog circuitry that exists in virtually all digital systems. Some of that circuitry is now showing up in application-specific ICs. It is expected that by 1990, half of all semicustom circuitry, and one third of standard-cell designs will be more than 10 percent analog [Goer]. Roughly 80 percent of all printed circuit boards contain some analog components today.

It appears, however, that even sophisticated users of design automation tools prefer to separate the analog and digital portions of their systems; the analog circuitry is generally verified using some derivation of the popular SPICE circuit simulator [Coh, Vlad], and there are many commercially available logic simulators that can be used for simulation of digital circuits [VLSI]. Unfortunately, it is rather difficult to correlate these two distinct simulations, and to analyze how analog and digital parts affect each other. Clearly, an "integrated" approach is needed which can handle both analog and digital simulation within one, consistent simulation environment.

Three basic approaches have been taken toward mixed analog-digital simulation. The first method uses an analog simulator to perform both analog and digital simulation [Getr, ORou]; the second uses a digital simulator to perform both digital and analog simulation [SaDi]; in the third method two simulators, a digital and analog one, are coupled together [CoWi]. In the first method, the digital elements are usually analyzed by the same mechanisms as the analog ones, which results in too accurate but also too inefficient simulation. The second approach extends the digital (discrete) methods to analog elements; this usually provides quite efficient but rather inaccurate simulation. Only the coupled approach combines the advantages of analog (accuracy) and digital (efficiency) simulations but this depends upon the level of coupling. Loosely coupled simulators basically execute two independent simulation programs (analog and digital) that "communicate" whenever they need information from the other part of the circuit; they are relatively simple to design but perform simulation of mixed analog–digital circuits neither really accurately nor efficiently. Tightly coupled or integrated simulators "synchronize" the two simulation mechanisms at the level of internal timesteps and time event control, so they can easily avoid any redundant evaluations without any loss of accuracy.

Any implementation of integrated mixed-mode simulation must solve two basic questions [Samp], (i) conversion of analog to digital and digital to analog information on interfaces of analog and digital components, and (ii) synchronization of the (usually variable) timesteps of the analog simulation [McC,Pede] with the event list that drives the (event-driven) digital simulation [MME]. The analog-to-digital conversion can be handled by establishing voltage thresholds and corresponding digital signals (the conversion is performed by elements called "thresholders" [MAR]). The digital-to-analog conversion is more difficult because it must generate a continuous analog waveform on the basis of discrete digital values. Two popular simple solutions assume that the converted waveforms are piecewise linear and piecewise exponential. A more sophisticated conversion has been implemented for example in SAMSON [SaDi].

This paper describes an approach to integrated mixed-mode simulation in which analog simulation is provided by SPICE-PAC [Zub1], a modified SPICE simulator. SPICE-PAC is a simulation package that is upward compatible with the popular SPICE simulation program [Coh,Vlad]. It means that SPICE-PAC accepts the same circuit description and performs all the analyses which are available in the SPICE programs, but also provides a number of features that do not exist in SPICE, for example an access to internal values of circuit elements, hierarchical naming scheme, parameterized subcircuit expansion, dynamic definitions of parameters and outputs, and also "enhanced" circuit simulation in which users can extend or modify some standard

simulation capabilities by their own routines in order to increase efficiency, accuracy or applicability of simulators. Digital simulation capability is an example of such enhancements.

In mixed-mode simulation, analysis of digital circuits is performed either by user-defined simulation routines (at any level of abstraction), or is performed by a "standard" digital simulator with a flexible time control mechanism needed for synchronization with the analog simulation algorithm. General organization of mixed-mode simulator is shown in Fig.1.

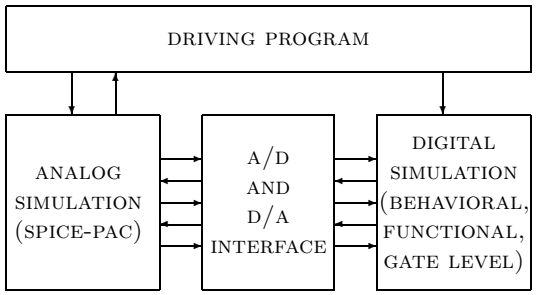

Fig.1. Organization of mixed-mode simulation.

The paper is organized in three main sections. Section 2 describes modified time-domain analysis, extended by an interface to digital simulation. Section 3 presents extensions to the input language for specification of analog/digital and digital/analog interfaces, it also describes the interfacing routines in some detail. Simulation of the digital part of a circuit is discussed in section 4.

2. MODIFIED TIME-DOMAIN ANALYSIS

SPICE-PAC, similarly to the SPICE program, implements the time-domain (transient) analysis in two consecutive steps, the so called "Initial Transient" analysis and the proper "Transient" analysis. Initial transient analysis finds the "initial" solution (i.e., the solution of circuit equations for the zero time instant) with user-specified or default initial conditions. The proper transient analysis integrates the differential equations describing the circuit from one timepoint to another, using either the trapezoidal rule (default) or backward differentiation (or Gear's) method [Coh,Vlad].

The proper time–domain analysis is controlled by two mechanisms, (i) the variable timestep which is determined by user specified parameters as well as truncation errors estimated during integration steps, and (ii) the so called "breakpoint table", BPT, which contains all "characteristic" time instances of time-dependent source functions as well as contributions introduced by transmission lines.

The variable timestep is controlled by iteration count and an estimated truncation error [Coh]. The iteration count uses the number of Newton-Raphson iteration steps 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 maximum timestep determined by the value of DELMAX. DELMAX (TMAX in [Vlad]) is either specified as one of parameters of time-domain analysis, or (by default) DELMAX = (Time_stop - Time_start)/50, where "Time_stop" and "Time_start" are parameters of time-domain analysis (TSTOP and TSTART in [Vlad]). Furthermore, if the number of iterations is greater than the limit ITL4, the Newton-Raphson iteration is terminated as nonconvergent, the timestep is divided by 8, and the iterative solution begins for a new timepoint, provided the reduced timestep is greater than the minimum timestep, DELMIN $= 10^{-9}$DELMAX (otherwise the notorious message "internal timestep too small" is reported and the analysis terminates).

The estimation of (local) truncation errors determines the minimum timestep for which the truncation errors are "acceptable". When the truncation errors (for the assumed timestep) are "unacceptable", the integration backtracks in time, and the new, "acceptable" timestep is used provided it is greater than the minimum timestep DELMIN.

In SPICE-PAC, the original implementation of the time-domain analysis has been extended by invocations of an auxiliary routine `digital` for simulation of digital parts of circuits. These additional invocations are within the initial section of analysis, within the iterative integration scheme, and in the error termination sequence. `digital` is an (internal) interfacing routine, which returns the "time" of the next event `TNEXT` if the digital simulation is event-driven (otherwise `TNEXT` is zero). `digital` is discussed in greater detail in the following section.

```
initialize;
TIME:=0;
update_time_dependent_sources(TIME);
solve_the_initial_system_of_circuit_equations;
create_breakpoint_table(BPT);
DELTA:=Time_step;
DELBKP:=DELTA;
DELMIN:=1D-9*DELMAX;
if (mixed-simulation) then
   call digital(initial_parameters);
IBRTAB:=1;
BREAKP:=true;
error:=false;
while (not error) do
   if (mixed_simulation) then
   {  call digital(current_parameters,TNEXT);
      if (reset) then mixed_simulation:=false;
      if (terminate) then Time_stop:=TIME;
      if (TNEXT>0) then
         if (TNEXT<TIME+DELTA) then DELTA:=TNEXT-TIME;
   store_the_solution;
   if (TIME > Time_stop) then
      interpolate_output_results_and_return;
   if (BREAKP) then
   {  IBRTAB:=IBRTAB+1;
      DELTA:=
         min(DELTA,0.1*min(DELBKP,BPT[IBRTAB]-TIME));
      if (IBRTAB=2) then DELTA:=0.1*DELTA;
      BREAKP:=false }
   else if (TIME+DELTA > BPT[IBRTAB]) then
   {  DELBKP:=DELTA;
      DELTA:=BPT[IBRTAB]-TIME;
      BREAKP:=true };
   call solve_timepoint;
endwhile;
if (mixed_simulation) then
   call digital(termination_parameters);
stop_analysis(error_termination);
```

"BREAKP" is a logical flag that is used in coordination of the variable timestep DELTA with the BPT; its "true" value indicates that 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 "special" changes of voltages and/or currents at the breakpoint.

It should be observed that the digital simulation is invoked before the `store_the_solution` operation as the digital simulation may change the timepoint (iterating a threshold value of an analog-to-digital signal, as shown in the next section).

The procedure `solve_timepoint` is as follows:

```
timepoint:
 TIME:=TIME+DELTA;
 if (ITL5>0 and total_number_of_iterations>ITL5) then
    set_error("limit of iteration steps reached")
 else if (execution_time > time_limit) then
    set_error("execution time limit reached")
 else
 {  update_time_dependent_sources(TIME);
    solve_the_system_of_circuit_equations;
    if (converged) then
    {  DELOLD:=DELTA;
       estimate_integration_errors_and_adjust(DELTA);
       if (integration_error_is_acceptable) then
          return;
       TIME:=TIME-DELOLD }
    else
    {  TIME:=TIME-DELTA;
       DELTA:=DELTA/8 };
    BREAKP:=false;
    if (DELTA >= DELMIN) then go to timepoint };
 set_error ("timestep too small");
 return;
```

The `solve_timepoint` procedure is also used in analog-to-digital conversion when "threshold timepoints" are iterated.

3. INTERFACE TO DIGITAL SIMULATION

A basic analog-digital interface implemented in SPICE-PAC provides a table-driven conversion of analog to (multi-valued) digital signals and vice versa. Piecewise linear characteristics of independent voltage and/or current sources [Coh, Vlad] are used for interactions between digital and analog subnetworks; the "smoothing" of discrete digital signals is thus implemented by piecewise linear functions. The interface is composed of two sections, one for analog-to-digital communication, and the second for communication in the opposite direction. In the input (circuit specification) language, these two sections are described by two new directives, PUTLIST and GETLIST, respectively:

```
.PUTLIST:Tname1 Voutput1,Voutput2,...
.GETLIST:Tname1:Tname2 Vsource1,Vsource2,...
```

where "Tname1" indicates a TABLE pseudoelement [Zub2] that defines the conversion table for analog-to-digital (and digital-to-analog) interface; "Tname2" indicates another TABLE pseudoelement that defines the delay table for digital-to-analog conversion; each "Voutput" is a voltage output in the SPICE sense, i.e., it is either "V(node1,node2)" or "V(node1)" if the second node is zero; each "Vsource" is the name of an independent voltage source with a piecewise linear time-dependent function.

The conversion table is defined as an ordered sequence of increasing (threshold) voltages interposed with (internal) values of corresponding digital signals:

```
.TABLE Tname Volt0 Num1 Volt1 Num2 Volt2 ... Numk Voltk
```

The digital equivalent of voltages in the range "Volt0" to "Volt1" is a signal represented by the value "Num1", etc. For digital-to-analog conversions, the extreme values "Num1" and "Numk" are translated into "Volt0" and "Voltk", respectively, while all intermediate values are converted into "median" voltages, i.e., "Num2" corresponds to "(Volt1+Volt2)/2", etc.

In the present implementation, the delay table contains just three parameters, the delay time of the converted digital-to-analog signals (indicated in GETLIST), the rise rate (i.e., the rise time per 1V) and the fall rate.

In the following example, the digital part (not shown here) is a two-input one-output block, with inputs indicated by the PUTLIST and the output by GETLIST. It should be observed that there are two different conversion tables for analog-to-digital and digital-to-analog conversions, and also the rise and fall rates are different:

```
VV 1 0 PULSE(-5.0,+5.0,0.5US,10NS,10NS,2US,5US)
R1 1 2 1K
C1 2 0 1NF
R2 1 3 1K
C2 3 0 200PF
VX 5 0 PWL(0 -5.0,15U -5.0)
RX 5 0 1K
.TRAN 50NS 10US
.PRINT TR V(2) V(3) V(5)
.PUTLIST:TCONV1 V(2),V(3)
.GETLIST:TCONV2:TDEL VX
.TABLE TCONV1 (-5.0,-1,-1.0,1,+5.0)
.TABLE TCONV2 (-5.0,-1,+1.0,1,+5.0)
.TABLE TDEL (2E-7,5E-8,2E-8)
.END
```

The condition `mixed_simulation` used in the modified time-domain analysis (previous section) is satisfied only if both PUTLIST and GETLIST are nonempty, and then the interfacing routine `digital` is invoked for each successfully solved timepoint. The interfacing routine performs analog-to-digital conversion of all PUTLIST voltages, and then checks if if any digital value created during this conversion differs from the "previous" value, and if all digital values remain the same, if the present timepoint has been explicitly requested by the digital simulation routines (for example, because of the "internal" timing mechanisms). If both checks fail, the interfacing routine terminates, and the time-domain analysis continues otherwise the digital simulation is performed. If any one of the PUTLIST signals changes its (digital) value, before invocation of the digital simulator the timepoint is (iteratively) adjusted to a value corresponding to the closest analog-digital conversion threshold.

Digital simulation is performed either by a "standard" digital (or logic) simulator, or by a user-defined routine which analyzes the digital part at the gate, functional or

behavioral level. After completion of digital simulation, the digital-to-analog conversions are performed for all those (digital) signals which are indicated in the GETLIST specifications and which change their (digital) values during the simulation.

The outline of the interfacing routine is as follows:

```
digital_simulation:=false;
min_threshold:=max_value;
for i:=1 to length_of_PUTLISTs do
{  extract_the_voltage(i,voltage);
   convert_analog_to_digital(voltage,signal);
   if (signal <> old_input_signal[i]) then
   {  digital_simulation:=true;
      find_the_closest_conversion_threshold(voltage,x);
      if (x < min_threshold) then
      {  min_threshold:=x;
         n:=i };
      old_input_signal[i]:=signal }};
if (digital_simulation) then iterate(n,min_threshold)
else if (TIME=next_event_time) then
        digital_simulation:=true;
perform_one_step_of_digital_simulation;
for j:=1 to length_of_GETLIST do
{  extract_the_signal(j,signal);
   if (signal <> old_output_signal[j]) then
   {  update_the_voltage_source(j,signal);
      old_output_signal:=signal }};
check_pending_events(TNEXT);
```

and the `iterate(n,threshold)` procedure performs a simple but robust K-step iteration (K is a parameter) in which the `old_input_signal` is now the "new" value of the corresponding signal:

```
Dmin:=0.0;
Dmax:=DELTA;
tol:=epsilon*abs(threshold+epsilon)
i:=0;
while i < K do
{  TIME:=TIME-DELTA;
   DELTA:=(Dmin+Dmax)/2;
   call solve_timepoint;
   extract_the_voltage(n,voltage);
   if (abs(threshold-voltage) < tol) then return
   convert_analog_to_digital(voltage,signal);
   if (signal=old_input_signal[n]) then Dmax:=DELTA
   else Dmin:=DELTA;
   i:=i+1 };
```

`epsilon` is another parameter that determines relative accuracy of "threshold iteration"; its default value is 0.05 .

4. DIGITAL SIMULATION

Digital simulation is performed (or controlled) by a routine called SPUSIM, which is either a simple interfacing routine to a "standard" logic simulation program [MME, VLSI], or a user-supplied routine (that "drives" or enhances a logic simulator). Its definition must be consistent with the following (FORTRAN) header:

```
SUBROUTINE SPUSIM (TIME,LINP,NINP,LOUT,NOUT,MARK)
DOUBLE PRECISION TIME
INTEGER LINP(NINP),LOUT(NOUT),MARK
```

where the parameters are:

`TIME` - the value of the actual (simulated) time,

`LINP` - an array of length `NINP` which contains the converted values of PUTLIST data,

`NINP` - the length of `LINP`, i.e., the number of analog-to-digital signals,

`LOUT` - an array of length `NOUT` which return the new (digital) values of GETLIST variables; on entry, `LOUT` contains "previous" values of GETLIST variables, so only changes need to be stored in `LOUT`,

`NOUT` - the length of `LOUT`, i.e, the number of digital-to-analog signals,

`MARK` - an entry/return flag; on entry: `MARK=-1` indicates the initial invocation, `MARK=0` indicates an accepted timepoint (i.e., a "regular" invocation), while nonconvergence (and termination of analog simulation) is indicated by `MARK=+1`; on exit: `MARK=0` indicates continuation of analysis, and `MARK=+1` a request to terminate the analysis at the current timepoint.

The following (Fortran) SPUSIM routine simulates a two-input XOR (exclusive-OR) gate and prints a trace of all invocations:

```
      SUBROUTINE SPUSIM (TIME,LINP,NINP,LOUT,NOUT,MARK)
      DOUBLE PRECISION TIME
      DIMENSION LINP(NINP),LOUT(NOUT)
      IF (LINP(1).EQ.LINP(2)) THEN
        LOUT(1)=-1
      ELSE
        LOUT(1)=+1
      ENDIF
      WRITE(NROUT,500) ATIM,LINP(1),LINP(2),LOUT(1)
500   FORMAT(' ... *spusim* :- time :',1PD9.2,'  inp ',
     +       2I3,'  out ',I3)
      RETURN
      END
```

When it is used with the analog circuit shown previously (section 3), it produces – at the analog and digital side of the interface – waveforms shown in Fig.2 and Fig.3.

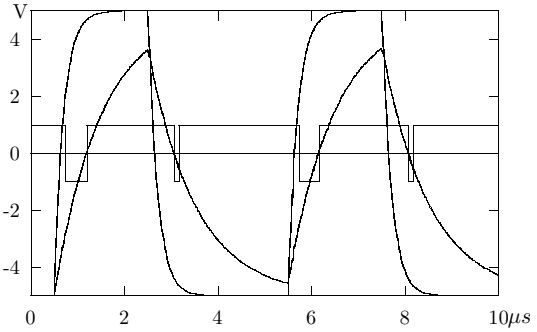

Fig.2. Analog PUTLIST and digital GETLIST waveforms.

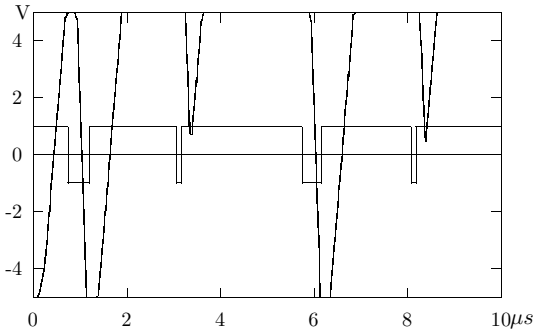

Fig.3. Analog and digital GETLIST waveforms.

It also generates the trace which is shown below:

```
... *spusim* :- time : 0.00D+00  inp  -1 -1  out  -1
... *spusim* :- time : 6.08D-07  inp  -1  1  out   1
... *spusim* :- time : 1.52D-06  inp   1  1  out  -1
... *spusim* :- time : 2.70D-06  inp   1 -1  out   1
... *spusim* :- time : 3.43D-06  inp  -1 -1  out  -1
... *spusim* :- time : 5.61D-06  inp  -1  1  out   1
... *spusim* :- time : 6.22D-06  inp   1  1  out  -1
... *spusim* :- time : 7.70D-06  inp   1 -1  out   1
... *spusim* :- time : 8.59D-06  inp  -1 -1  out  -1
```

If a typical gate-level simulator is used for digital simulation, the description (as a section of the input data) may look like:

```
input : x1,x2;
output : y;
begin
   y:=xor(x1,x2)
end
```

where the "input" signals `x1` and `x2` correspond to consecutive (analog) PUTLIST elements, and the "output" signal `y` controls the GETLIST voltage source.

5. CONCLUDING REMARKS

The basic interface has been used to simulate a number of mixed analog-digital circuits, and in particular several A/D converters. In all cases a significant reduction of simulation times with respect to "all analog simulation" has been observed (in the range of one to two orders of magnitude depending on the level of digital simulation, behavioral, functional or gate-level). However, the development process of such simulations is quite unreliable and time-consuming because an independent simulation code (SPUSIM routines) has to be developed for each application, and this code must be tested and validated before actual simulations. Therefore, a "better" approach is needed, in which the digital simulation is derived directly from design specifications. This can be provided by a new "flexible" digital (multilevel) simulator that is being developed specifically for integration with SPICE-PAC.

The modifications presented in this paper are implemented in SPICE-PAC versions 2G6c.90 and beyond.

Acknowledgement

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

R e f e r e n c e s

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

[CoWi] T. Corman, M.U. Wimbrow, "Coupling a digital logic simulator and an analog circuit simulator"; VLSI Systems Design, vol.9, no.2, pp.38-47, 1988.

[Getr] I. Getreu, "Analog modeling language spans all system design levels"; Electronic Design, vol.35, no.13, pp.95-104, 1987.

[Goer] R. Goering, "A full range of solutions emerge to handle mixed-mode simulation"; Computer Design, vol.27, no.3, pp.57-65, 1988.

[MAR] H. de Man, G. Arnout, P. Reynaert, "Mixed-mode simulation techniques and their implementation in DIANA"; in: "Computer Design Aids for VLSI Circuits", P. Antognetti, D.O. Pederson, H. de Man (eds.), Sijthoff and Noordhoff 1981.

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

[MME] S. Murai, H. Matsushita, K. Enomoto, "Logis simulation programs"; in: "Logic design and simulation", E. Hoerbst (ed.), pp.135-163, North-Holland 1996.

[ORou] R. O'Rourke, "Behavioral modeling of digital devices in an analog simulation environment"; VLSI Systems Design, vol.9, no.1, pp.16-25, 1988.

[Pede] 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.

[SaDi] K.A. Sakallah, S.W. Director, "SAMSON : a mixed circuit-logic level simulator"; in: "Advances in Computer-Aided Engineering Design - vol.1", A. Sangiovanni-Vincentelli (ed.), pp.149-223, JAI Press 1985.

[Samp] G. Sampson, "An ideal mixed-mode simulator"; VLSI Systems Design, vol.9, no.11, pp.70-78, 1988.

[Vlad] 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.

[VLSI] VLSI Systems Design Staff, "1988 survey of logic simulators"; VLSI Systems Design, vol.9, no.2, pp.50-61, 1988.

[Zub1] W.M. Zuberek, "SPICE-PAC 2G6c - An Overview"; Technical Report #8903, Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1C 5S7, 1989.

[Zub2] W.M. Zuberek, "Enhanced controlled sources as device models in the SPICE-PAC simulation package"; Proc. 30-th Midwest Symp. on Circuits and Systems, Syracuse NY, pp.603-606, North-Holland 1988.