Unconventional Simulation tasks in OrCAD PSpice via Simulation Manager

MILAN JAROŠ[1], JAROSLAV KADLEC[2], DALIBOR BIOLEK[2,3]
[1] iSEC – IT Services and Enterprise Communications, Inc.,
[2] Dept. of Microelectronics, Brno University of Technology
[3] Dept. of EE, University of Defense, Brno, Czech Republic

Abstract: The paper describes a conception of the so-called simulation manager that considerably extends the application range of the OrCAD PSpice simulation program. It enables operating this program in the so-called sequential mode, when the relatively independent tasks are run consecutively with a possibility of data exchange. A powerful programming language also enables iterative runs within the conditional loops, which can be utilized e.g. for optimization.

Keywords: - PSpice, simulation, analysis, circuit.

1 Introduction

Programs of the Spice or PSpice type are widely used for solving various problems in electrical engineering both at academic institutions and in industry [1]. OrCAD PSpice [2] is one of today's well-known programs from this category. In contrast to the WinSpice, ISSpice4 and other programs, it does not support the utilization of ICL (Interactive Command Language) [3] for controlling the simulation tasks. However, this language represents a powerful tool for operation in the so-called sequential mode, when the simulation tasks are run consecutively, with the ability to influence the character of consecutive operations, depending on the attained state of the simulation run. That is why the OrCAD PSpice users cannot solve problems of the following character: Successive automated runs of different types of analyses, e.g. AC, Transient, DC, immediately after the end of the foregoing analysis, utilizing data from this analysis for the current simulation run. The consecutive modifications of model parameters, which would depend on the results of previous analyses. Repeated run of various simulation tasks in the loops until the optimal behavior of circuit model is reached.

The simulation manager (SiM), described in this paper, is designed to control the OrCAD PSpice in agreement with the user's intentions [4]. The controlling algorithm is defined by the so-called manager control file (MCF). This file should be written according to the syntactic rules of special programming language of the manager. This language contains, among other things, the instructions for compiling the ECIR (Extended Circuit File), which is a source text for generating the PSpice circuit file (PCIR), commands for defining the variables, for defining the basic PSpice analyses which should be executed, for controlling the PSpice operation, and for receiving the simulation results, saving them in variables, and processing them mathematically.

2 Conception of the simulation manager

The conception of the co-action of the SiM and the computing PSpice core is illustrated in Fig. 1.

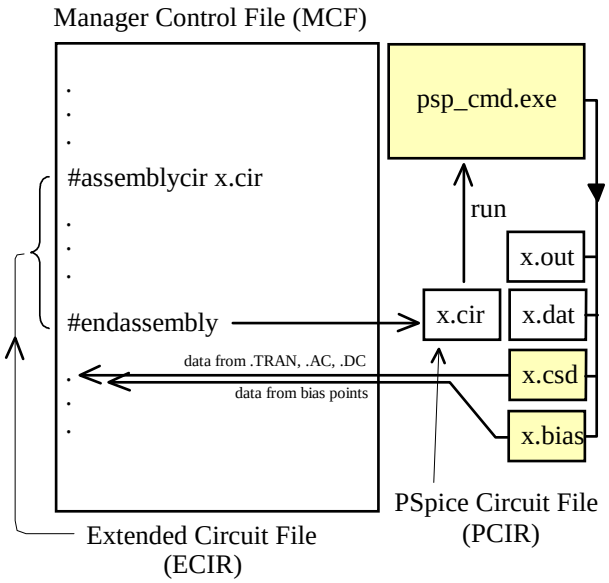

Fig. 1: Simplified schematic of the communication between the SiM and the simulation program via the Manager Control File.

The commands from the MCF are executed step by step in a sequence that is defined in this file. The source text of ECIR for generating the PCIR for running an independent simulation task is bounded by

a pair of the #assemblycir and #endassembly commands. In this text, the conventional PSpice syntax can be combined with the extended commands of the SiM. At the moment of processing the #endassembly command, the conventional PCIR is automatically generated, and the computational core psp_cmd.exe is subsequently run with the generated PCIR as a parameter. The manager is waiting until the end of simulation and then it finds out, on the basis of the return code, if the simulation ran correctly. In the case of an error, the operation of the manager is terminated. The user can identify this error from the output file generated.

When the simulation run is terminated correctly, both the output file and – depending on the character of the simulation task – other files containing the results of individual analyses (Transient, AC, or DC) are available, as well as the calculated coordinates of the bias points. All these results can be read as new variables of the SiM. For example, we define the *VP* variable and save the value of the voltage between the nodes p and 0 in time 10ms to this variable. These variables can be used for defining the PCIR of subsequently executed simulation.

3 Characteristic of the simulation manager and its language

The MCF serves as the input of the SiM. This file contains models of simulated circuits in the PSpice language and special commands for controlling the simulation tasks for the SiM. The commands for SiM should be unambiguously distinguishable from the PSpice commands. The structure of the MCF should be clear and transparent.

The MCF is thus a record of a program for the SiM. It was necessary to design a new, simple programming language for the above control of simulation tasks. The requirements for this language were defined as follows:

- The SiM will read the MCF step-by-step, starting from the first line (the MCF is processed sequentially).
- The programming language of the SiM should support simple mathematical computations. The program should therefore be able to work with variables representing real numbers and to evaluate arithmetic terms which can contain variables, numerical constants, basic operators (+, -, *, /, ^), braces, and some mathematical functions. In other words, some commands should be defined which enable the definition/declaration of variables and the evaluation of arithmetic terms.
- The ECIRs of the circuits being analyzed can appear in the MCF. Thus commands should exist for the definition of the beginning and the end of such an ECIR. The command defining the beginning should have a parameter indicating the file name. The given PCIR will then be generated into this file. The command defining the end of the ECIR will cause the PSpice to run with the name of the PCIR as a parameter. The SiM will wait for the end of the simulation and then it will continue on the next line of the MCF.
- The PCIR can be modified prior to its generation by the SiM. Recording the value of arithmetic term in a certain place of the PCIR is one of the alternatives. It should be possible to write in the text of the ECIR a command for evaluating the arithmetic term. At the moment when the PCIR is generated this term is evaluated by the SiM and its numerical value is written into the PCIR. This method can, for example, modify the parameters of some circuit components in order to perform optimization.
- The SiM should be able to process the results of executed simulations. Thus some commands should be defined for reading such results. The results of certain simulations can be saved by PSpice in the text files. The SiM should be able to process these files. The commands in the MCF will enable saving such values into user-defined variables. It will enable subsequent work with these values.
- After finishing the activities of the SiM, all the files generated from all executed simulation runs should be available. They are all PCIRs and the corresponding output files, and the data files containing the results of the simulations which were generated by utilizing the .SAVEBIAS or .PROBE commands. The SiM can perform the conservation of these files such that it will perform their backup under modified names after finishing the simulation run. The new file names will be derived from their original names via their extension by numbers corresponding to the order of the simulation run.
- The SiM should include commands for program loops and chaining (the if and while commands, known from other programming languages) on the

basis of boolean relations. By means of boolean relations it should be possible to compare the values of arithmetic terms, as well as to link the boolean expressions to more complicated units via logical operators (and, or, negation, etc.).

The commands for program loops and chaining should be also placed in the text for generating the PCIR. In this way, the user can control which parts of the PCIR will be generated or which parts will be repeated more times.

The *goto* command can also be included among the commands for program loops and chaining. It serves for passing the control to the given label (which is defined by the *label* command). This whole group is called "commands for run control". Thanks to these commands, we can algorithmize the evaluation of the results of foregoing simulations and control other simulations.

- A command should exist for including a file, analogous to the PSpice command *.INC*. The included file could also contain the commands of the SiM, thus it would be an MCF. That is why the PSpice command *.INC* cannot be used for such cases. Using this method, the program controlling the SiM could be divided into several files, making its structure more transparent. We can also reuse the already generated parts of the program by this approach.
- The possibility of writing the comments belongs to the natural demands on the SiM language. A reasonable choice is to assume the same format as in PSpice: if the first character on the line is *, then the content of the line means a comment. When the comments are written in the ECIR which generates the PCIR, they will be also written into this PCIR.

The SiM works as an interpreter of the above language. The SiM operation can be divided into two phases. The syntactic analysis of the MCF is performed in the first step. Syntactic errors can be found here, i.e. an incorrect notation of the SiM commands or their incorrect location. The response to this error means terminating the MCF processing and displaying an error message.

During the syntactic analysis, the text of the MCF is fragmented into individual syntactic elements (e.g. commands, comments, lines of the generated PCIR, etc.). The syntactic elements are represented by a data structure (class). A list of these elements or commands is created during the syntactic analysis. The second phase of the SiM operation consists in performing the commands from this list. The SiM activity is terminated if the end of this list is achieved or if a command for breaking or stopping is executed or in the case of error (e.g. division by zero, etc.).

In the course of processing the MCF, the syntax correctness of generated PCIRs is not checked. This checking will be only done by PSpice itself when processing this PCIR. Then PSpice will simultaneously carry out the simulation.

The result of syntactic checking or the occurrence of another error can be learned from the return code of the psp_cmd.exe program. When the return code is zero, the simulation was carried out without error. If not, some error appeared. A description of this error is given in the output file. If such an error occurs during the simulation, the SiM is terminated.

4 Demonstration of working with SiM

A schematic of transistor amplifier with the stabilization of operating point via negative feedback is in Fig. 2.

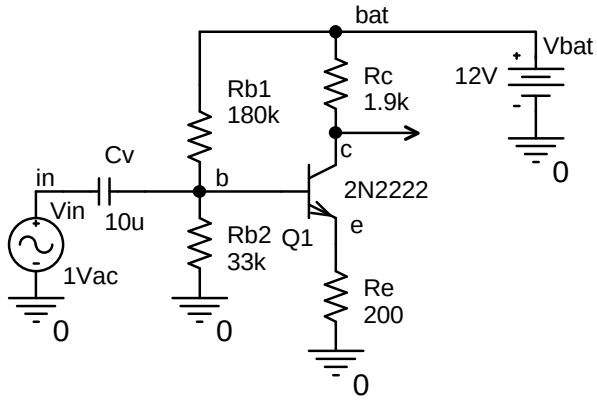

Fig. 2: Example of the simulated circuit. R_c is to be designed such that the voltage gain is 10 on a frequency of 1kHz.

The PSpice analysis shows that the AC gain of the circuit within the amplifier passband is about 9. The task is to design Rc such that the gain is 10 on a frequency of 1kHz.

A complete content of the MCF is listed below. For lucidity, the individual lines are numbered. Note that these numbers are not part of the MCF.

```
1:      *transistor amplifier
2:      #defsim AC .AC dec 1 1k 10k
3:      #set Rc 1.9k gain 1
4:      *
5:      #while (gain)<=10
6:      #assemblycir run.cir
7:      *
8:      Vbat bat 0 12V
```

```
9:    Q c b e Q2N2222
10:   Rc bat c #$Rc$
11:   Re e 0 200
12:   Rb1 bat b 180k
13:   Rb2 b 0 33k
14:   Cv in b 10u
15:   Vin in 0 AC 1
16:   .lib
17:   #runsim AC
18:   .print AC v([c])
19:   #endassembly
20:   #setprobe AC V([c])
21:   #getprobe gain AC V(c) 1k
22:   #set Rc Rc+20
23:   #endwhile
```

The first line represents a conventional header of the circuit file according to common PSpice conventions. The SiM commands, starting with the symbol #, are placed on lines 2-6. The simulation type labeled AC is defined on line 2. It is a basic PSpice AC analysis which is written on this line in accordance with common PSpice syntactic rules. Here it is a single-point AC analysis at a frequency of 1kHz. The *Rc* and *gain* variables are declared by the #set command on line 3, and the numerical values are assigned to them. It is an analogy to the PSpice *.PARAM* statement. The *while* loop starts on line 5 with its end on line 23. Its purpose is to test if the *gain* variable is less than or equal to 10. If yes, the instructions within the loop body are performed, else the activity of the SiM is terminated, because line 23 is the last line of the MCF.

The loop body starts by the definition of the ECIR (the #assemblycir command on line 6 with the pair command #endassembly on line 19). The PCIR is generated according to this definition. The conventional PSpice netlist of the amplifier from Fig. 2 is written on lines 8 to 15 with one exception on line 10, where the formula #Rc is used for the definition of resistance *Rc*. The # symbol means that the interpretation of this formula will be provided by the SiM. Then the pair characters $ $ follow, between them is the formula. The numerical value of this formula is included into the generated PCIR by the SiM. In this case, the formula is very simple because it contains only the *Rc* variable.

The command for performing the analysis, defined on line 2, is written on line 17. The difference between them and the conventional PSpice .AC command consists in the fact that here the analysis results are stored automatically into the data file for the PROBE postprocessor, but in the text format *CSDF*, which can be read by the SiM. A detailed setting of the parameters of *.PROBE* command can be defined by the #setprobe command on line 20. According to this concrete setting, only values of voltages at node *c* will be saved to the data file. The PSpice *.PRINT* command is added to line 18; it saves the AC voltage of the node c into the output file. This voltage is equal to the value of AC gain of the amplifier. The #getprobe command on line 21 saves the above value into the *gain* variable. Then the #set command on line 22 increases the value of Rc by 20 Ohms. Both the generation of the PCIR run.cir and its analysis are repeated inside the *while* loop until the *gain* exceeds the value 10.

In this case, the analysis runs 11 times. One can find the following pairs of the *Rc* and *V(c)* variables in last two output files:

2100 Ohms, 9.997,
2120 Ohms, 10.09,

thus the optimal value of *Rc* lies inside the interval of (2.1 - 2.12) kOhms.

5 Conclusions

The simulation manager (SiM), described in this paper, is an independent executable program which enables, with the utilization of the so-called Manager Control File (MCF), an effective control of the OrCAD PSpice program. Currently the design of the console-type SiM is completed, which works on the text file level. Simultaneously we are developing a graphical user's interface which enables comfortable programming of sequential operations also for users who do not need to master the script language of the SiM.

Acknowledgment

This work is supported by the Grant Agency of the Czech Republic under grant No. 102/08/0784, by the research programmes of BUT MSM0021630503, MSM0021630513, and UD Brno MO FVT0000403.

References

[1] Vladimirescu, A. 'The SPICE book', John Willey&Sons, Inc., 1994.

[2] PSpice Reference Guide. Cadence Design Systems, Inc.

[3] Láníček, R. 'Simulation programs for Electrical Engineering'. BEN – technical literature, 2000.

[4] Jaroš, M. 'Simulation manager for SPICE-compatible programs', Bachelor's Thesis, UMEL FEKT VUT Brno, 2007 (in Czech).