IEEE Int. Symposium on Circuits and Systems (ISCAS'94), London, England, 30 May - 2 June 1994, pp.33-36.

An Approach to Integrated Numerical and Symbolic Circuit Analysis

W.M. Zuberek
Department of Computer Science
Memorial University
St. John's, Canada A1C–5S7
tel: (709) 737–8627
wlodek@cs.mun.ca

A. Konczykowska
CNET – PAB – MCT
Laboratoire Bagneux
92220 Bagneux, France
tel: (33)(1) 4231-7462
aga@bagneux.cnet.fr

Abstract

An interface to symbolic circuit analysis has been developed for a SPICE–like circuit simulator in order to integrate numerical simulation with an existing program for symbolic circuit analysis. In effect, both numerical and symbolic analyses use the same internal representation of circuits which makes the two approaches truly complementary. This integrated simulation capability is used in simulation–based parameter extraction where all ac small-signal parameters are fitted through the symbolic analysis rather than numerical one, significantly reducing the execution time of the extraction process.

INTRODUCTION

Increasing demand for electronic components results in rapidly growing both the scale and the size of electronic circuits. This continuously creates needs for new and more efficient analog methods for circuit analysis. Integration of numerical and symbolic circuit analyses is one of possible improvements than can be used to increase the efficiency of circuit analysis tools.

Reliable computer–aided circuit analysis or circuit simulation cannot be obtained without accurate specification of circuit elements and device models. Existing device models use large sets of parameters, values of which must be properly determined to represent device characteristics accurately. Because of highly nonlinear device models, these parameters usually cannot be determined by direct measurements; popular extraction methods use iterative techniques to minimize differences between measurement data and model behavior in the full range of operating conditions.

One of flexible approaches to parameter extraction is to use a circuit simulator rather than a set of model equations (such an approach is called simulation–based parameter extraction). An important advantage of the simulation–based method is that the extractor can use all the capabilities of the circuit simulator, so all packaging and mounting parasitics can easily be taken into account during extraction, and the extraction can use many types of measurement data, including noise, distortion, etc. On the other hand, repeated simulations can easily become rather time–consuming, especially when numerous parameters are extracted from large sets of measurement data. In some cases (e.g., parameter extraction for microwave applications), groups of parameters correspond to linear analyses of the circuit. For linear analyses, the dependence of circuit responses on some variables can be derived in a symbolic form, and this symbolic form can be used very efficiently in repeated analyses of the same circuit for different combinations of values of the variables. Therefore, a circuit simulation package used for simulation–based parameter extraction (the FIT program developed at CNET) has recently been enhanced by an interface to symbolic circuit analysis. The interface is composed of a number of functions which provide a convenient access from the symbolic analyzer to the internal representation of the analyzed circuit. The input module of an existing symbolic analyzer has been modified to conform to the interface from a SPICE-like numerical simulator. The results of the symbolic analysis, i.e., the symbolic functions, can be generated in several forms, depending upon application. One of such applications, integrated numerical–symbolic parameter extraction, is discussed in greater detail in this paper.

The paper first very briefly outlines the basic ideas of numerical and symbolic circuit analysis, and then discusses symbolic simulation in the context of parameter extraction. It also outlines the implementation of the interface between the numerical and symbolic simulators. A simple comparison of execution times required for numerical and symbolic simulations of a typical circuit used in parameter extraction, and for several sets of data, is included.

NUMERICAL AND SYMBOLIC SIMULATION

The popular 'third–generation' (numerical) circuit simulators [5] use a modified form of nodal analysis (modified to take care of voltage sources, floating sources, and inductive elements) and Newton–Raphson iteration to solve the system of simultaneous nonlinear algebraic equations

$$F(X) = 0$$

which describes the balance of currents at the nodes of the network in terms of node voltages (and some branch currents) X. The solution is typically obtained through a sequence of linear approximations to the nonlinear function $F(X)$ at points $X^{(j)}$

$$G(X^{(j)})\Delta^{(j)} = -F(X^{(j)})$$

where G is the Jacobian of F with respect to X (evaluated at $X^{(j)}$), and $X^{(j+1)} = X^{(j)} + \Delta^{(j)}$. The iteration terminates when $\Delta^{(j)}$ is sufficiently small.

This basic scheme is used in the DC operating point, DC transfer curve, and even time–domain analysis; in the last case, the dependence upon time is eliminated by approximating the differential equations by difference equations [4, 6]. Only frequency–domain (small–signal) analyses are significantly different because they require (for each frequency) a solution of a system of simultaneous linear equations in the complex domain; this is often done by separating the real and imaginary parts of coefficients and variables, and solving a twice as large system of linear equations in the real domain.

The 'main computational effort' of numerical circuit simulation in typical applications is thus devoted to: (i) evaluating the Jacobian G and the function F, and then (ii) solving the system of linear equations.

The principle of symbolic simulation [1, 3] is to derive analytic (or symbolic) network functions from a representation of a network rather than solve (numerically) the systems of circuit equations. This means that (some of) circuit parameters are represented by symbols in the derived functions, and then the circuit responses can be obtained very efficiently by evaluation of the derived analytic formulas.

Symbolic simulators use different circuit representations and different algorithms to derive network functions. The algorithm used in the integrated numerical/symbolic simulator (called FIT–S) uses the Coates flowgraph representation. Variables corresponding to graph nodes are the same as those used in the modified nodal analysis. The characteristic functions are in the form of rational functions:

$$H(x) \;=\; \frac{N(x)}{D(x)} \;=\; \frac{\sum_i x^i A_i(s_1, ..., s_m)}{\sum_j x^j A_j(s_1, ..., s_m)}$$

in which the numerator $N(x)$ and the denominator $D(x)$ contain coefficients $A(s_1, ...s_m)$ which are (nested or expanded) polynomial functions in symbolic elements $s_1, ..., s_m$.

In fully expanded form, the polynomial coefficients are in the 'sum–of–product' form:

$$A(s_1, ..., s_m) \;=\; \sum_{i=1}^{k} C_i \prod_{j=1}^{n} S_{ij}$$

where C_i are real numbers, S_{ij} are circuit symbols, and k and n depend upon the flowgraph's structure (which reflects the topology of the circuits).

SYMBOLIC SIMULATION IN PARAMETER EXTRACTION

For parameter extraction in general, but especially in the case for microwave applications, a significant part of the extraction process analyzes the small–signal, linear behavior of the circuit. These linear analyses can conveniently be performed using symbolic simulation rather than numerical one, and obtaining circuit responses very efficiently from symbolic functions. Moreover, it is often the case that the extraction of a set of parameters is decomposed into a sequence of 'partial extractions', performed on subsets of parameters and relevant subsets of measurement data [2]. For such partial extractions, the sets of parameters are usually quite small which means that the corresponding symbolic functions are also quite simple.

For simulation–based parameter extraction (as implemented in the FIT program) the (iterative) frequency–domain analyses are performed within a general optimization scheme:

```
while continue_optimization do
   update_the_values_of_parameters;
   error_value := 0;
   for each frequency_domain_data_group do
      update_op_point_voltages_and_currents;
      find_the_operating_point_solution;
      for each frequency do
         find_the_solution_of_linear_equations(res);
         update(error_value,data,res)
      endfor
   endfor;
   for each non_frequency_domain_data_group do
      find_circuit_responses(res);
      update(error_value,data,res)
   endfor;
   get_new_values_of_parameters(error_value)
endwhile
```

Since all frequency–domain analyses are performed for the circuit with the same topology, the generation of symbolic functions can be done only once. Furthermore, the number of of symbols which can change their values during one optimization cycle (or one 'partial extraction') is rather small, and includes a subset of extracted parameters (updated in the optimization loop) and all those symbols which depend upon the operating point solution. All such symbols are called *variable* symbols while the remaining symbols are called *fixed* symbols. It should be observed that all *fixed* symbols can be replaced by their numerical values during the generation of the symbolic functions, reducing the functions and simplifying the subsequent evaluations. The symbolic functions at this stage can thus be represented by

$$\mathcal{F}_i \;=\; x^{k_i}\mathcal{P}_i \sum_{j=0}^{n_i} x^j \mathcal{R}_{ij}$$

where each $\mathcal{P}_i$ is a product of a constant C_i and (some) symbols S_{ik}, $k = 1, ..., m_i$

$$\mathcal{P}_i \;=\; C_i \prod_{k=1}^{m_i} S_{ik}$$

and each $\mathcal{R}_{ij}$, $j = 0, 1, ..., n_i$, is a sum of products

$$\mathcal{R}_{ij} \;=\; \sum_{k=1}^{\ell_{ij}} C_{ijk} \prod_{\ell=1}^{m_{ijk}} S_{ijk\ell}$$

Moreover, the values of *variable* symbols can be retrieved in two steps: (i) at the beginning of the optimization loop for extracted parameters, and (ii) after each operating point solution for symbols which depend upon operating–point (i.e., small–signal parameters). These values of variable symbols are used for transformation of the generated symbolic functions to the reduced form

$$\mathcal{F}_i^{(r)} \;=\; x^{k_i} A_i \sum_{j=0}^{n_i} x^j A_{ij}$$

where all A_i and A_{ij}, $j = 0, 1, ..., n_i$, are constants provided that no frequency–dependent elements are used. Only this very simple polynomial form needs to be evaluated in the innermost (i.e., frequency) loop.

Symbolic simulation can be included in the previous optimization scheme in the following way:

```
retrieve_the_values_of_all_fixed_symbols;
generate_symbolic_functions;
while continue_optimization do
   update_the_values_of_parameters;
   retrieve_the_values_of_extracted_parameters;
   error_value := 0;
   for each frequency_domain_data_group do
      update_op_point_voltages_and_currents;
      find_the_operating_point_solution;
      retrieve_the_values_of_dependent_symbols;
      evaluate_coefficients_of_reduced_functions;
      for each frequency do
         evaluate_reduced_functions(val);
         convert_to_circuit_responses(val,res);
         update(error_value,data,res)
      endfor
   endfor;
   for each non_frequency_domain_data_group do
      find_circuit_responses(res);
      update(error_value,data,res)
   endfor;
   get_new_values_of_parameters(error_value)
endwhile
```

the step `generate_symbolic_functions` generates the products $\mathcal{P}_i$ and $\mathcal{R}_{ij}$ above, and the step `evaluate_parameters_of_reduced_functions` calculates the values of A_i and A_{ij} using the retrieved values of variable symbols.

INTEGRATION OF NUMERICAL AND SYMBOLIC SIMULATION

Any integration of numerical and symbolic simulations must provide some sort of interaction between these two types of analyses. In the FIT–S program, the interaction is performed through an interface which supports the following operations (implemented as interface procedures):

- RESET(NAME),
- NEXTEL(DESC,TYPE,NODES,LEN),
- GETVAL(DESC,TYPE,VALUES,LEN).

RESET must always be used as the first operation, before any other operation of the interface; it initializes extraction of circuit elements for symbolic analysis; its parameter tt name is either '*' which indicates all elements of the simulated circuits, or it must be a name of a subcircuit expansion (i.e., an `X`–name in the SPICE convention) which indicates the subcircuit for symbolic analysis.

NEXTEL returns the descriptor DESC, the type TYPE, and the list of nodes NODES of length LEN) of the next circuit element (or indicates that the 'next' element does not exist); it is implemented in such a way that consecutive invocations of this operation return descriptions of consecutive circuit elements (according to the internal representation of the circuit); zero returned as the value of DESC indicates that there are no more elements.

GETVAL uses the vector VALUES to return the numerical value(s) of parameters associated with an element identified by DESC and TYPE; LEN is set to the number of values returned in VALUES.

Typical sequence of interface operations (during generation of symbolic functions) is as follows:

```
reset(name);
graph := 0;
nextel(desc,type,nodes,num);
while desc > 0 do
   add_to_flowgraph(id,nodes,graph);
   add_to_symbol_table(desc,type,id);
   if fixed_symbol(desc,type) then
      getval(desc,type,value,len);
      store_in_symbol_table(id,value)
   endif;
   nextel(desc,type,nodes,num)
endwhile;
generate_symbolic_functions(graph);
```

The symbol table combines all attributes of all symbols used in the simulation and extraction. These attributes include the class of symbols (fixed, extracted, dependent), which is used for selective retrieval of values. For example, after each solution of the operating point, the values of all operating-point dependent symbols are retrieved from the circuit description and stored in the symbol table:

```
for each symbol in symbol_table do
   if dependent(symbol) then
      getval(desc,type,value,len);
      store_in_symbol_table(id,value)
   endif
endfor;
```

The table of symbols is used during evaluation of all coefficients A_i and A_{ij} of reduced functions.

COMPARISON OF NUMERICAL AND SYMBOLIC SIMULATION

The comparison of numerical and symbolic simulation is given for parameter extraction of a submicron (0.25 μ) GaAs FET on InP substrate. A small-signal model (with its parameters) is shown in Fig.1.

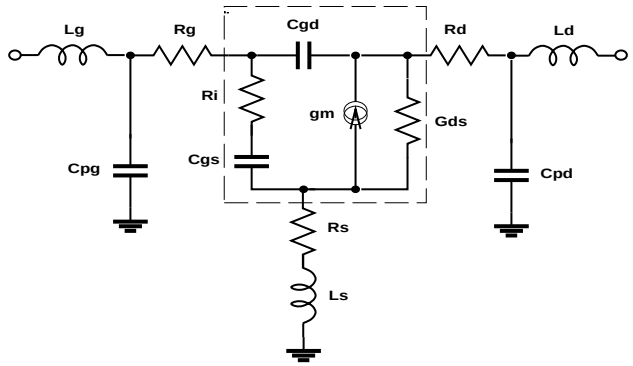

Fig.1. GaAs FET small-signal model.

For typical small–signal models the number of nodes is usually less than 10, and the number of symbols is less than 15.

For the small-signal model shown in Fig.1, the (five) reduced symbolic functions are polynomials of degrees 3, 6, 7, 5 and 6. Since a polynomial of degree n can be evaluated using $n-1$ multiplications and n additions, the number of operations (additions and multiplications) required for evaluation of all reduced functions is of the order of 50 (although the polynomials are in the complex domain, they are evaluated in the real domain). Unfortunately, the evaluation of the symbolic functions is only a rather insignificant part of all computations involved in parameter extraction; the values of symbolic functions must be converted into S–parameters, they must be stored in a database of data, compared with the corresponding measurement values to update the value of the error function, etc. A more realistic comparison of symbolic and numerical analysis is thus obtained by measuring the total execution times for typical extraction data. These results are summarized in the following table in which the columns correspond to data groups with 10, 20, 50 and 100 frequency values (the execution times are in seconds, on a SPARCstation 2, for 20 iteration steps):

exec time	10	20	50	100
symbolic	0.64	0.72	0.85	1.12
numerical	1.72	2.50	4.66	8.32
speedup	2.7	3.5	5.5	7.4

The number of variable symbols influences the execution time rather insignificantly, especially for data groups with large number of frequency values; the number of variable symbols affects the evaluation of the coefficients of reduced functions, but this evaluation is performed only once for each data group.

CONCLUDING REMARKS

An integration of symbolic simulation with traditional numerical methods can significantly reduce the simulation time. In the case of parameter extraction, this reduction can stimulate the use of more sophisticated extraction strategies, which – in general – are more computationally demanding.

In the case of parameter extraction, the analyzed circuits are rather small, so the symbolic functions are relatively simple and no function approximations are really needed. More general applications of symbolic and integrated numerical/symbolic simulations must take into account that for larger circuits the symbolic functions become very complex, so additional function simplification is required [1].

The speedup factor of the symbolic approach depends upon a number of implementation issues, for example, the representation of symbolic functions, factorization, and other reduction and optimization techniques. Further speedup can be obtained if the implementation is 'tuned' properly.

REFERENCES

[1] G. Gielen, W. Sansen, "Symbolic analysis for automated design of analog intergrated circuits"; Kluwer Academic Publ. 1991.

[2] A. Konczykowska, W.M. Zuberek, J, Dangla, "Parameter extraction with the FIT–2 program"; Proc. European Conf. on Circuit Theory and Design, Copenhagen, Denmark, pp.762–771, 1991.

[3] P-M. Lin, "Symbolic network analysis"; Elsevier 1991.

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

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

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

[7] W.M. Zuberek, A. Konczykowska, C. Algani, H. Wang, J. Dangla, "Simulation–based parameter extraction, its implementation and some applications"; IEE Proc. on Circuits, Devices and Systems (in print), 1994.