A Graph Based Symbolic Analyser for the Characterization of Analog Circuits and the Automatic Generation of Design Equations

M. Fakhfakh, M. Loulou and N. Masmoudi
Laboratoire d'Electronique et des Technologies de l'Information.
National Engineering School of Sfax, Tunisia.
fakhfakhmourad@lycos.com

Abstract

A symbolic analyser is presented. It automatically generates analog circuit design equations. It is based on the use of directed graph theory. Needed expressions, transfer functions, poles and zeros and others can be automatically get from a given net list. A schematic support is used for this purpose.

Index Terms--- Symbolic analyser, graph theory, automatic circuit design.

1. Introduction

Design of analog circuits generally relies on both designer experience and the use of a numerical simulator [1]. However these approaches are nevertheless possible for the design of complicated circuits. The task rapidly becomes hard, tedious and error prone since the formulation of needed equations is hand made. In addition, designer can not have direct and easy information on components forming poles and zeros, and, obviously, can not act on them.

For these reasons, symbolic analysers are, nowadays, at the aim of interest of analog designers [1, 2]. Indeed, such tools are receiving strong attention because they allow and facilitate design of complicated circuits (involving transistors, dependent sources, etc.), and study of the behaviour of a circuit. Furthermore, they form an essential complement to numerical simulations [3, 4].

In this paper, we present a tool that we developed for this purpose. It automatically generates symbolic and/or semi symbolic design equations. It is based on the use of directed graph theory. In fact, we choose the graph approach for two reasons:

- Firstly, representing a circuit by its correspondent directed graph allows designer to have right information on his circuit's main characteristics [1, 5]. So with a minimum experience on handling this tool, he becomes able to introduce the right modifications on this circuit (for example; to modify circuit's characteristics by adding extra components in the right place),

- Secondly generation of symbolic expression is easier when compared to other approaches. Besides, it allows to ease the problem of the maximum analysed circuit size and memory limitation (especially for matrix transformations).

2. Architecture of the developed tool

Usually, determination of circuit design equations is based on the use of KCL and KVL (*Kirchoff* Current and Voltage Laws). Generally obtained systems are written in matrix form [3]:

$$A.X = B \tag{1}$$

where $A = [a_{ij}]_{nxn}$ is the coefficient matrix, and $X^T = [x_1, x_2, \ldots, x_n]$ and $B^T = [b_1, b_2, \ldots, b_n]$ refer to the symbolic circuit parameters and the input vector respectively.

If the coefficient matrix is non singular, the solution of (1) can be written as follows:

$$x_k = \frac{\sum b_i \Delta_{ik}}{\det(A)} \tag{2}$$

where Δ_{ik} denotes the correspondent cofactor.

However this representation is not adequate. Indeed, since automated generation of design equations is needed, designer have not to make any mathematic formulation [5,6].

Fortunatly, diagrammatic representations offer interrest spare solution. Thus, we use directed graph theory to solve equations such as (2).

A useful link between a determinant of a matrix and its corresponding graph is obtained by referring to the defnition of a determinant:

$$\det(A) = \sum_i \alpha_{i1i2\ldots in} \; a_{i1}\, a_{i2} \ldots a_{in} \tag{3}$$

where $\sum_i \alpha_{i1i2\ldots in} \; a_{i1}\, a_{i2} \ldots a_{in}$ is the sum taken over all permutations, $\alpha_{i1i2\ldots in}$ is equal to –1 or +1 if $i_1 i_2 i_3 \ldots i_n$ is an odd or an even permutation respectively.

In our symbolic tool we use this feature to determine dependency between all circuit edges, admittances of loops, dependency between the graph loops. Also, paths between input node and the specified output vertex are calculated in the same way (i.e. by the use of the matrix permanent developped formula).

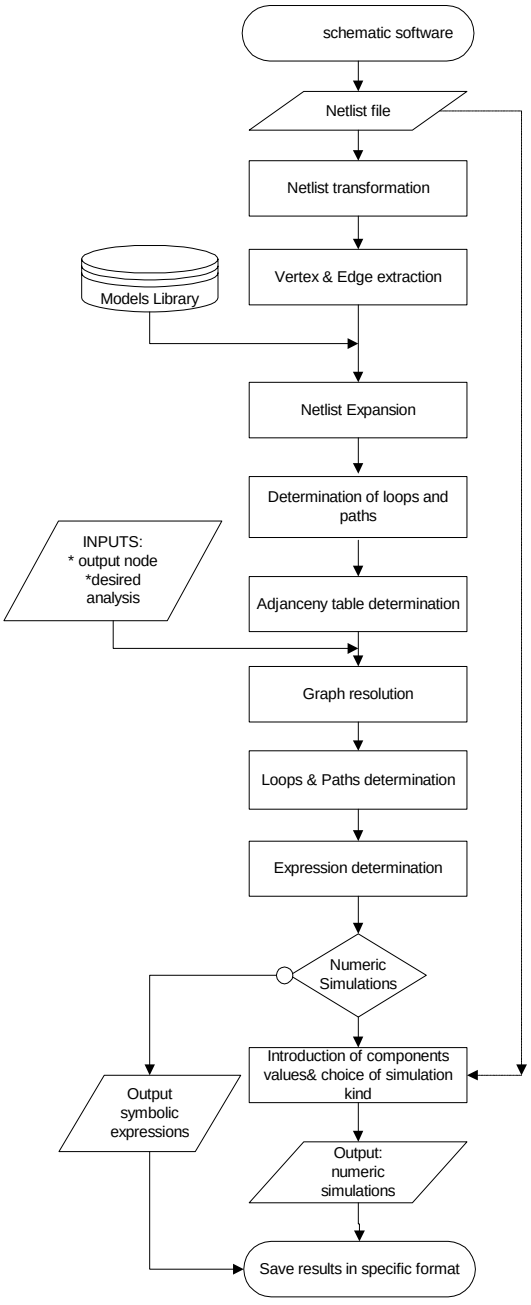


fig.1: flowchart of the symbolic tool architecture

The architectural diagram of the symbolic analyser tool is depicted at fig 1. The flow of main operations in this software can be summarized as follows:

- From a schematic software, a net list is obtained (we use any schematic software for this purpose),
- A software program written in MATLAB software, reads the net list, converts it to a suitable form for the graph treatment, does the chosen analyses and returns result in a symbolic form as given by (4):

$$TF = \frac{\sum_{i=0}^{m} x^{i} s^{i}}{\sum_{j=0}^{n} x^{j} s^{j}} \qquad (4)$$

where s is *Laplace* operator, and x refers to circuit's parameters. TF refers to the transfer function.

Semi-symbolic analysis is also possible, since dependency between the circuit behaviour and one (or more) component is sometimes needed. Besides, a series of simulations can be obtained by variation of the values of a 'generic' component in a specified range.

- Simulations are also possible. Component values are, therefore, introduced or get from the net list file (it depends on the simulation kind the designer looks for; simple or parametric).

3. Application Examples

Two simple examples are detailed below to highlight some of the abilities of the symbolic analyser (of course more complicated circuits can be easily treated, however, their transfer functions will take much place to be presented).

- *Example n°1:*

An example of an analog filter is depicted at fig.2. Net list, obtained from the schematic software, is given at table I. Figure 3 depicts the directed graph representation of the circuit illustrated at fig.2. we put the stress on the fact that the graph is programmed in the symbolic software and the obtained result is directly given without any need of presenting the graph.

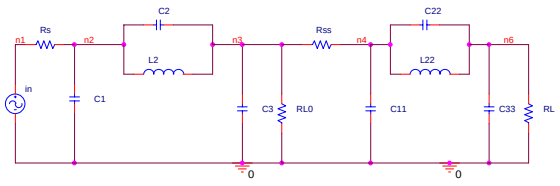


fig.2: An example of analog filter with state variables

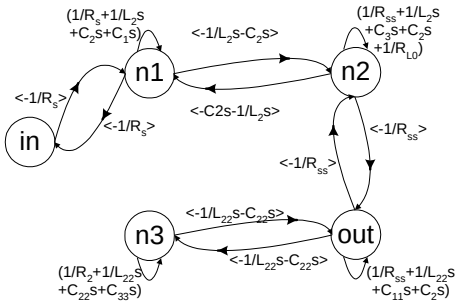


fig.3: Graph representation of the circuit of fig. 2.

The symbolic transfer function (TF) obtained thanks to the symbolic Analyser tool is given below. We see clearly how complicated this TF is, even though many simplification considerations are taking into account. Simplifications are done just to get a reasonable number of terms for the proposed example, other ways TF contains more than 600 terms. *(Rs=RL0=R0;Rss=RL=R;C3=C1=C2=C0;C11=C22=C33:=C;Rs=RL:=Ri; L2=L22=L.)*

```
@ source FILTER
R_Rs          N1 N2  1
V_entree      N1 0 DC 1V AC 1V
C_C1          0 N2  1.44483
R_RL0         0 N3  1
C_C2          N2 N3  0.20718
L_L2          N2 N3  0.93666
C_C3          0 N3  1.44483
C_C11         0 N4  1.44483
L_L22         N4 N6  0.93666
R_RL          0 N6  1
C_C22         N4 N6  0.20718
C_C33         0 N6  1.44483
R_Rss         N3 N4  1
```

Table 1 : Netlist of the analog filter depicted at fig.1

*TF:=(2*C0*L^4*C*Ri^2*R*s^6+C0*L^4*R*Ri*s^5+(2*C*L*Ri+C0*L*Ri)*L^2*R*Ri*s^4+L^3*R*Ri*s^3+L^2*Ri^2*R*s^2)/(9*L^4*Ri^3*C0^2*C^2*R^2*s^8+L^2*Ri*(12*C0*L^2*Ri^2*C^2*R+6*C0^2*L^2*Ri^2*R*C+6*C0^2*L^2*Ri*C*R^2+6*C^2*L^2*R^2*C0*Ri)*s^7+L^2*Ri*(3*C0^2*L^2*Ri*R+12*C0*L^2*Ri*C*R+4*C0*L^2*Ri^2*C+6*C^2*L^2*R*Ri+4*C*L^2*C0*R^2+6*C0*Ri^2*C^2*L*R^2+6*C0^2*L*Ri^2*R^2*C)*s^6+L^2*Ri*(4*C*L^2*R+2*C0*L^2*Ri+2*L^2*C0*R+2*L^2*C*Ri+6*Ri^2*C^2*L*R+8*C0*Ri*C*L*R^2+3*C0^2*L*Ri*R^2+12*C0*L*Ri^2*R*C+3*C0^2*L*Ri^2*R+3*C^2*L*R^2*Ri)*s^5+L^2*Ri*(10*C*R*L*Ri+2*C*L*R^2+2*Ri^2*L*C+2*C0*L*Ri^2+L^2+2*C0*R^2*L+8*C0*R*L*Ri+4*Ri^2*C0*R^2*C)*s^4+L^2*Ri*(2*Ri*C0*R^2+4*Ri^2*C*R+2*L*Ri+2*R^2*C*Ri+3*L*R+2*Ri^2*C0*R)*s^3+L^2*Ri*(3*Ri*R+Ri^2+R^2)*s^2)* (5)*

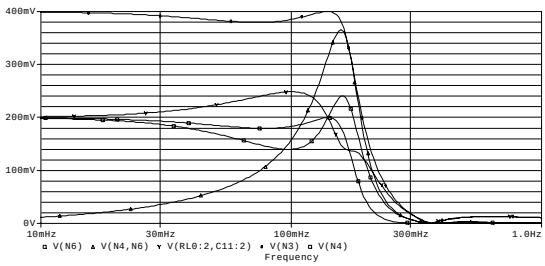


fig.4:SPICE Simulations

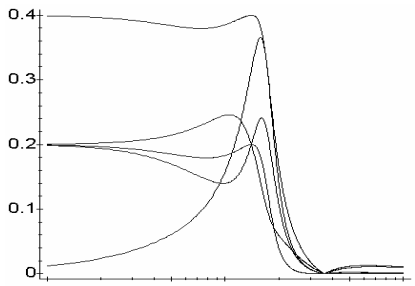


fig.5: MAPLE Simulations

Figure 4 illustrates numeric simulations obtained by SPICE software. At fig.5 we give output voltages obtained at the same nodes that those of fig.4: *n3*, *n4* and *n6*. These results are obtained by simulation, with MAPLE software, of transfer functions calculated at these nodes.

We can easily see the perfect correspondence between both simulations.

- *Example n°2:*

At fig.6. is depicted a classical MOS mirror circuit. Let TF the transfer function between output voltage (node 5) and input current *I0*. Its symbolic expression is given below. Fig.7 illustrates the graph representation of the circuit shown at fig.6.

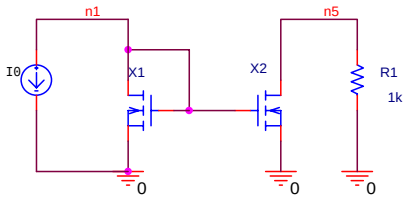


Fig.6: Simple MOS Current Mirror

*TF:=gmX2*R1/((CgsX1+CgsX2)*(g0X2*R1+1)*s+(g0X1+gmX1)*(g0X2*R1+1)* (7)

where *CgsXi*, *g0Xi* and *gmXi* are grid to source capacitance, output conductance and transconductance parameter of *Xi* MOS transistor .

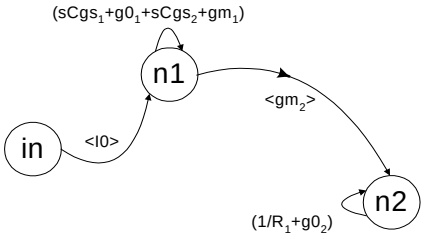
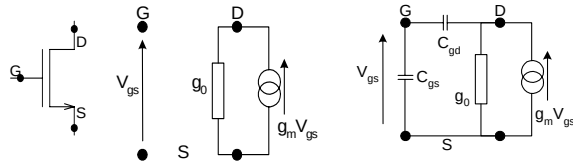


Fig.7: Graph representation of the circuit of fig. 6.



-a- -b- -c-

Fig.8: Two MOS transistor models

where g0, gm, Cgs and Cgd refer to output conductance, MOS transconductance, grid to source capacitance and grid to drain capacitance respectively.

We put the stress on the fact that the designer can choose his transistors level of modelling according to the needed precision. For MOS transistors three models can be chosen from the library. Two of them are depicted at fig.8-a and 8-b. In the third model fig.8-c, bulk node is taken into consideration.

4. Conclusion

In this paper, we present a symbolic analyser tool programmed with MATLAB software. The tool is based on the use of directed graph theory. The tool's conceptual and architectural approaches are detailed. The analyser allows the automatic featuring and the generation of analog circuit design equations. It automatically generates specific symbolic transfer functions from net lists obtained from a schematic software. Poles and zeros can also be automatically extracted. From the circuit's graph, one can have a direct idea on components forming zeros and/or poles and can 'act' on them. Two examples are given to illustrate some of the ability of the analyser.

Now we work on using graph theory to generalize the application of this software to automatically design optimised MOS transistor circuits, such as OTAs, current conveyors and switched current memory cells,. We also project to introduce, in this tool, a procedure to simplify obtained functions by means of genetic algorithms.

5. References

[1] D. Narshingh, *Graph Theory With Applications to Engineer and Computer Science*. Prentice Hall, inc. 1974. ISBN 0-13-363473-6.

[2] R. Diestel, *Graph Theory*. Electronic Edition 2000. Springer Verlag New York 197-2000. ISBN 0-397-98976-5.

[3] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits*, Kluwer AcademicPublishers 1991. ISBN 0-7923-9161-6.

[4] M. Ismail and J. Farnca, *Introduction to Analog VLSI Design Automation*, Kluwer Academic Publishers 1990. ISBN 0-7923-9071-7.

[5] G. Gielen, P. Wambacq and W. Sansen, "Symbolic Analysis Methods and Applications for Analog Circuits: A Tutorial Overview", Proceeding of the IEEE, Vol 82, n°2, February 194.

[6] .N. Horta, M. Fino and J. Goes, "Symbolic techniques Applied to Switched Current ADCs Synthesis", ISCAS 2000, IEEE International symposium on circuits and systems, May 28-31, 2000, Geneva, Switzerland.