DDD-Based Symbolic Sensitivity Analysis of Active Filters

A.A. Palma-Rodriguez[1,2], E. Tlelo-Cuautle[1], S. Rodriguez-Chavez[1,2], S. X.-D. Tan[2]
[1] INAOE. Department of Electronics. México.
[2] University of California at Riverside. Electrical Engineering. USA
{adairpalma, etlelo, srodriguez }@inaoep.mx, stan@ee.ucr.edu

***Abstract*—Sensitivity analysis helps circuit designers to determine boundaries to predict the variations that a particular design variable will generate in a target specification, if it differs from what is previously assumed. It is quite important in designing active filters, but the major drawback is the need of generating the symbolic transfer function, from which the normalized sensitivity formula is usually applied. In this paper we show the usefulness of applying determinant decision diagrams (DDDs), to compute sensitivities without generating the symbolic transfer function, previously. Basically, we propose replacing all symbols in the DDD graph by their numerical values, except one, i.e. the variable of interest. Therefore, all sensitivities are computed by applying *Cramer's* rule and by constructing one DDD for each symbol. We demonstrate the computing time gain compared to generating the fully symbolic transfer function and then applying the normalized sensitivity formula.**

I. Introduction

Sensitivity analysis is very important in integrated circuit (IC) design, because it helps us to optimize the behavior of a given circuit by showing us which components of the entire systems are more sensitive. Moreover, it can help us to reduce costs of production given that we can replace the less sensitive components with cheaper ones, and critical components with high quality components, and therefore more expensive [1] . Although symbolic circuit analysis is not popular because of its high computation cost, there has been a great progress by applying techniques such as Determinant Decision Diagrams (DDDs) [2]. That way, in this paper we propose a DDD-based method to compute sensitivities of analog ICs. We demonstrate that the appplication of DDDs leads us to improve symbolic sensitivity analysis by obtaining exact symbolic expressions from a nodal analysis formulation using nullor and pathological elements [3]. Moreover, it is worthy mentioning that the advantage of symbolic sensitivity analysis lies in the fact that is easier to compute the sensitivities with respect to many parameters, provided that a symbolic transfer function exist [4] . That improvement in performing DDD-based symbolic sensitivity analysis is possible since we reduce large symbolic expressions, given that we use circuits with a high number of transistors.

This paper is organized as follows: Section II describes an overview on DDDs. Section III describes some useful identities to accelerate the evaluation of the normalized sensitivity analysis formula. Section IV explains briefly the application of DDDs to compute sensitivities in analog ICs. Section V lists some active filters used as test circuits. Section VI shows the experimental results generated by our proposed DDD-based tool. Finally, Section VII summarizes the conclusions.

II. Determinant Decision Diagrams (DDDs)

The circuit size is a challenge in performing symbolic analysis because a large number of symbolic terms are manipulated. Fortunately, this problem is mitigated when applying a graph-based approach, e.g. Determinant Decision Diagrams (DDDs) [2], [3], [4], [5]. In fact, the DDD-based representation is compact for large class of analog circuits. Every determinant has a unique representation, and is amenable to symbolic manipulations [2], [5].

To understand how DDDs work, lets us consider the following determinant:

$$det(M) = \begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj - adhi - aefj - bcgj + cbih \tag{1}$$

If the determinant is $n \times n$, we expect to have paths of n levels. So if there is a path that is not complete, i.e. that does not have n levels or that has a zero in any element of the path, it will be eliminated completely since this means that this expression is multiplied by zero. As a result, for this particular example we obtain the DDD shown in Fig. 1, where all multiplications by zero were omitted.

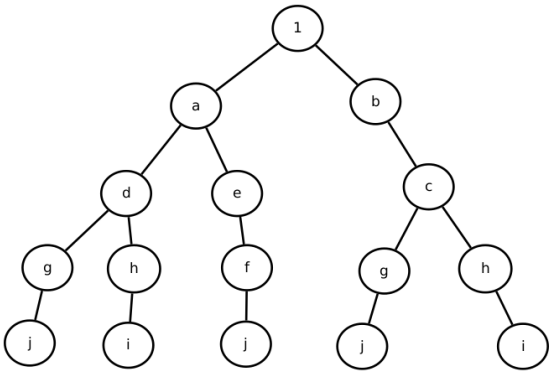

Fig. 1. Determinant Decision Diagram of (1).

978-1-4577-1117-6/12/$26.00 ©2012 IEEE

As one sees, this DDD structure is built recursively in a depth-first fashion [2], [4], and path eliminations are performed by sending a prune signal if a zero is found inside the path. Furthermore, the symbolic determinant expression is obtained by adding leaves in the same levels and by multiplying the ones in different levels. That is, different depth in the tree implies multiplication, while equal depth implies addition. This leads us to get the expression in (2).

$$det(M) = a(d(gj - hi) + e(-fj)) + b(c(-gj + hi)) \quad (2)$$

This DDD-based method is aready introduced in [5]. A key point is related to the assignation of signs to each node in the DDD expansion. They are established by applying the rule of signs from Cramer's rule.

When applying DDDs to evaluate a determinant, not only one obtains a factorized exact symbolic expression, but also one can derive all transfer relationships with respect to each node, and in a postprocessing step to each branch circuit variable.

III. Sensitivity Analysis

Circuit sensitivity can be defined as how much a particular circuit characteristic changes as a particular circuit-component value varies. In this investigation, we are working with the computation of AC sensitivities in active filters. For instance, one of the more popular notions used in circuit design is the adjoint network analysis [6], also implemented in the circuit simulator SPICE. The drawback of using SPICE to obtain the sensitivity of an analog IC with respect to a given circuit element, is that one has to execute AC sensitivity, then calculate the finite difference and apply normalization to get the numerical sensitivity. Afterwards, one need to plot the real and imaginary parts versus the frequency ω to generate the sensitivity curves. This is the main reason why we propose to apply our proposed DDD-based tool to compute symbolic sensitivities of analog ICs. In fact, DDD-based approaches can also be applied to compute symbolic sensitivities of large analog ICs in a hierarchical fashion [7].

In general, given a transfer function $H(s)$ of the form:

$$H(s) = \frac{N(s)}{D(s)} \quad (3)$$

where both $N(s)$ and $D(s)$ can be represented by a DDD, the AC sensitivity evaluation is obtained by the following normalized equation [6]:

$$Sens(H(s), W) = \frac{W}{H(s)} \frac{\partial H(s)}{\partial W} \quad (4)$$

Besides, after algebraic manipulations, the following expression is most useful for developing computational procedures [7]:

$$Sens(H(s), W) = W \left(\frac{1}{N(s)} \frac{\partial N(s)}{\partial W} - \frac{1}{D(s)} \frac{\partial D(s)}{\partial W} \right) \quad (5)$$

IV. Symbolic Sensitivity Computation by DDDs

Equation (5) is quite suitable to perform symbolic sensitivity computation of analog ICs. For instance, one advantage in applying DDDs, is that in the resulting sum of symbolic-product-terms, one can derive each product with respect to the desired variable, directly. Moreover, that desired symbolic variable in the DDD can be replaced by 1 in the paths it is contained, while eliminating those paths that does not contain the symbolic variable. The key point is that the determinant is not evaluated, that process is performed in the determinant expansion, thus saving CPU time. For example, if we want to get the derivative with respect to h in (2), we will have the updated expanded DDD shown in Fig. 2 .

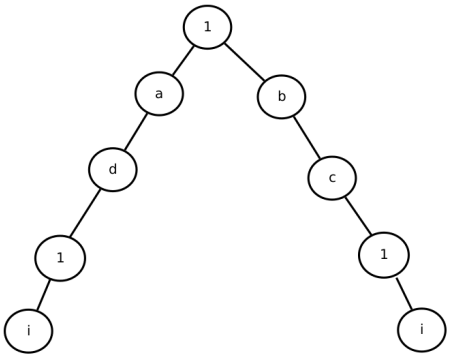

Fig. 2. Resulting graph by applying the derivative

In this manner, the derivative symbolic expression becomes (from (2)):

$$\frac{\partial det(M)}{\partial h} = adi + bci \quad (6)$$

Further, from (5), we still have to perform several operations to obtain the sensitivity of the transfer characteristic $H(s)$ with respect to a desired circuit element (W). This process is highly improved if we proceed as follows: Using the determinant described by (1), and its associated DDD shown in Fig. 1, suppose we want to realize the derivative with respect to g and we update the symbolic variable h to become $h = g + a$. The DDD representation of this example is shown in Fig. 3.

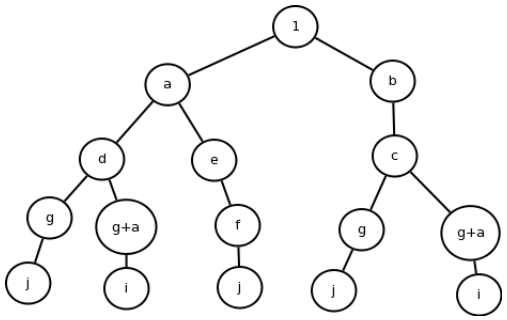

Fig. 3. Resulting graph by applying the derivative.

What we need to do is leave the value as it is if the node contains only one value, but if contains a sum of values, as the case of h, we should replace all elements by the variable of interest taking into account the associated sign. This is done only if the variable of interest is in the path, otherwise, the path is eliminated as explained before. The resulting DDD is shown in Fig. 4.

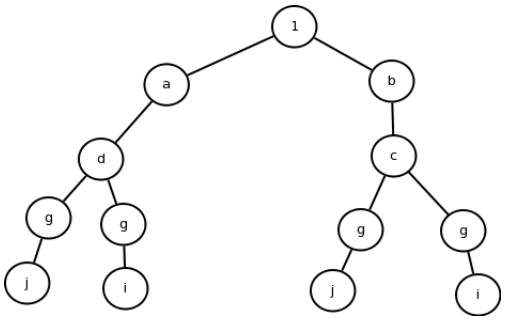

Fig. 4. Resulting graph by applying the derivative.

The symbolic expression by applying the derivative from Fig. 4 becomes:

$$g{\cdot}\frac{\partial det(M)}{\partial g} = adgj{+}adgi{+}bcgj{+}bcgi = adg(j{+}i){+}bcg(j{+}i) \tag{7}$$

As one sees, we can save CPU time for the multiplication step by applying this procedure.

V. Active Filters

In this paper we analyze the sensitivities of a current-mode KHN active filter, already presented in [8]. The goal is to obtain the sensitivies of all transfer characteristics with respect to all circuit elements R and C. The modified active filter is shown in Fig. 5, it consists of fully-differential current conveyors (FDCCIIs) instead of inverting CCIIs.

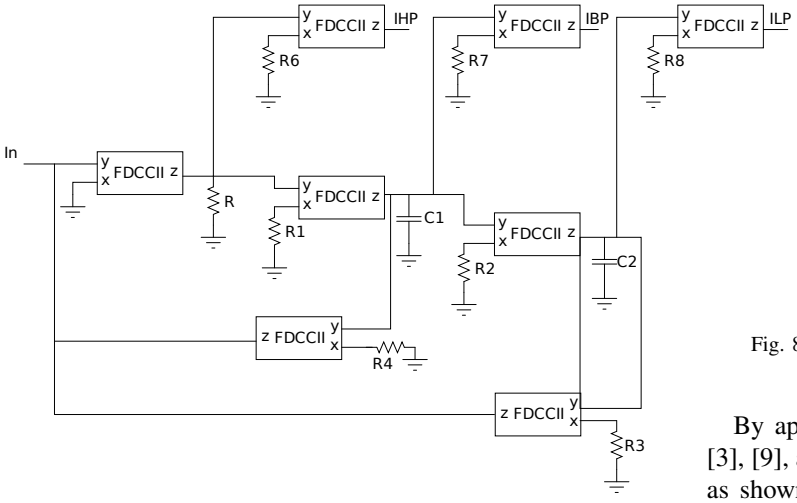

Fig. 5. KHN filter impemented with current conveyors.

Some properties of this active filter are that it has very low input impedance, employs grounded resistor and capacitors, and has independent control on the quality factor Q and on the gain. This KHN filter uses current conveyors whose nullor equivalent can be derived by applying the modeling approach introduced in [3], [9]. Additionally, we use another two active filters to compute their sensitivities. They are already presented in [10], and shown in Figs. 6 and 7.

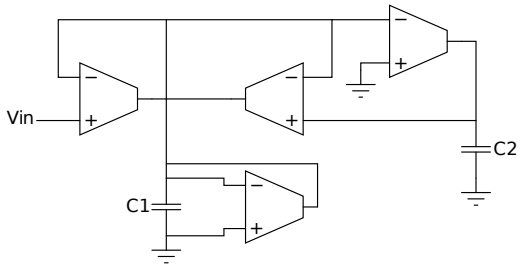

Fig. 6. Lowpass active filter with operational transconductance amplifiers.

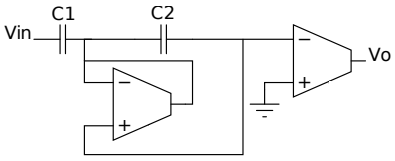

Fig. 7. Highpass active filter with operational transconductance amplifiers.

In any kind of active filter, not only it is desirable to derive a sensitivity expression with respect to every circuit element, but also it is desirable to include characteristics of the active devices. For example, the operational transconductance amplifier (OTA) used in this paper to implement the OTA-based active filters is shown in Fig. 8.

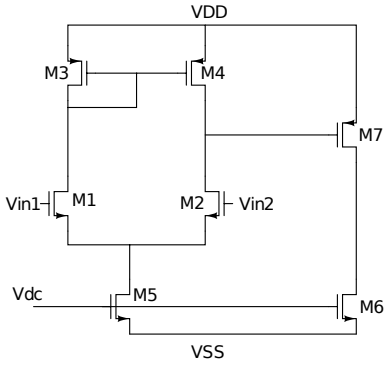

Fig. 8. Two-stage CMOS operational transconductance amplifier.

By applying the behavioral modeling method presented in [3], [9], a MOSFET can be represented by its nullor equivalent as shown in Fig. 9. It is important to notice that this model includes some dominant parasitic components. The nullor equivalent of Fig. 8 is shown in Fig. 10.

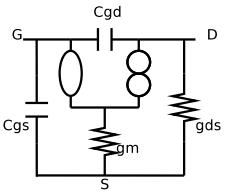

Fig. 9. Nullor representation of the transistor with parasitic elements.

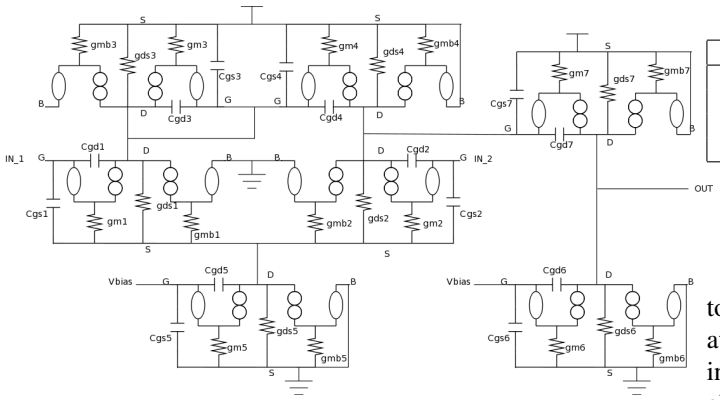

Fig. 10. Nullor equivalent of Fig. 8 including parasitic elements.

VI. Experimental Results

The sentivity computations were performed for the active filters shown before. The results obtained for the current-mode KHN active filter implemente with FDCCIIs are shown in Table I. We list the symbolic sensitivity expressions of one characteristic with respect to eight of the ten passive circuit elements.

TABLE I
KHN's Sensitivity computation results.

Element	Sensitivity Expression
R1	$-\frac{1.5625\cdot10^{35}}{(1.5625\cdot10^{13}\,\mathrm{R1})\,s^2-1.5625\cdot10^{23}\,s-1.9531\cdot10^{31}}$
R2	$-\frac{1\cdot10^{22}}{\mathrm{R2}\,s^2+(-1.25\cdot10^{6}\,\mathrm{R2})\,s-1.25\cdot10^{8}}$
R3	$\frac{-1.9531\cdot10^{31}\,\mathrm{R3}}{(1.5625\cdot10^{13}\,\mathrm{R3})\,s^2+(-1.9531\cdot10^{20}\,\mathrm{R3})\,s-1.9531\cdot10^{31}}$
R	$\frac{(1.5624\cdot10^{35})\,R}{-1\cdot10^{21}\,s^2+(1.5625\cdot10^{23}\,R)\,s+1.9531\cdot10^{31},R}$
R4	$\frac{-(1.5624\cdot10^{35})\,\mathrm{R4}}{(1.2500\cdot10^{17}\,\mathrm{R4})\,s^2-1.2500\cdot10^{28}\,s-1.9531\cdot10^{31}\,\mathrm{R4}}$
R8	$-\frac{1.5625\cdot10^{35}}{(1.2500\cdot10^{17}\,\mathrm{R8})\,s^2+(-1.5625\cdot10^{23}\,\mathrm{R8})\,s-1.9531\cdot10^{31}\,\mathrm{R8}}$
C2	$-\frac{1.5624\cdot10^{35}}{(1.25\cdot10^{28}\,\mathrm{C2})\,s^2+(-(1.5625\cdot10^{34})\,\mathrm{C2})\,s-1.9531\cdot10^{31}}$
C1	$-\frac{1.5624\cdot10^{35}}{(1.25\cdot10^{28}\,\mathrm{C1})\,s^2-1.5625\cdot10^{23}\,s-1.9531\cdot10^{31}}$

Results of the symbolic sensitivities computations for the Highpass and Lowpass active filters implemented with the OTA shown in Fig. 8, are presented in Table II. The computations were performed with and without including parasitics to the MOSFETs (Cgd, Cgs, gds, gm). It is important to notice that the number of circuit elements and the inclusion of parasitic elements increases much more the number of symbolic product-of-terms (labeled as expressions). For this small circuits we list the gain in CPU time (labeled as rate).

The total of symbolic sensitivity expressions which can be derived in this case, are equal to the number of circuit elements, which expressions are similar as the ones listed in Table I. This is the major advantage of our proposed DDD-based tool.

TABLE II
Symbolic sensitivity computation results.

Active filter	No. transistors	Expressions	Rate
Highpass(no parasitic elements)	16	21	1.5x
Highpass(parasitic elements)	16	63	2x
Lowpass (no parasitic element)	32	39	1.5x
Lowpass (parasitic elements)	32	123	1.5x

VII. Conclusion

In this paper we showed the usefulness of applying DDDs to compute symbolic sensitivity expressions, without generating the symbolic transfer function previously, as it is done in classical hand calculations. We proved that by replacing all symbols in a DDD except the variable of interest, the overall symbolic sensitivity analysis with respect to each circuit element becomes highly improved, while all sensitivity expressions can be derived efficiently.

VIII. Acknowledgments

This work is partially supported by UC-MEXUS and CONACYT under grants CN-11-575 and 131839-Y.

References

[1] Lucian Mandache, Michai Iordache and Luci Dumitriu, *Sensitivity and Tolerance Analysis in Analog Circuits using Symbolic Methods*, 10th International Conference on Development and Application Systems, pp. 230-235, Suceava, Romania, May 27-29, 2010.

[2] C.-J. Shi, S. X.-D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams", *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 1, pp. 1–18, 2000.

[3] C. Sánchez-López, F.V. Fernández, E. Tlelo-Cuautle, S. X.-D. Tan, "Pathological Element-Based Active Device Models and Their Application to Symbolic Analysis," *IEEE Trans on Circuits and Systems I*, vol. 58, no. 6, pp. 1382-1395, 2011.

[4] G. Shi, "Advances in Symbolic Techniques for Analog Design Automation", in *Analog Circuits: Applications, Design and Performance*. E. Tlelo-Cuautle (Ed.), NOVA science publishers, 2012.

[5] S. X.-D. Tan, "Symbolic Analysis by Determinant Decision Diagrams and Applications", in *Design of Analog Circuits through SYMBOLIC ANALYSIS*, Bentham Science Publishers Ltd. 2012.

[6] J. Vlach, K. Singhal, "Computer Methods for Circuit Analysis and Design", New York, NY: Van Nostrand Reinhold Company, 1983.

[7] X. Li, H. Xu, G. Shi, A. Tai, *Hierarchical Symbolic Sensitivity Computation with Applications to Large Amplifier Circuit Design*, in *Proc. International Conference on Circuits and Systems (ISCAS)*, Rio de Janeiro, Brazil, May 2011.

[8] A.M. Soliman, "Kerwin Huelsman Newcomb Filter Using Inverting CCII", *J. of Active and Passive Electronic Devices*, vol. 3, no. 3-4, pp. 273-279, 2008.

[9] E. Tlelo-Cuautle, C. Sánchez-López, D. Moro-Frias, "Symbolic analysis of (MO)(I)CCI(II)(III)-based analog circuits", *International Journal of Circuit Theory and Applications*, vol. 38, no. 6, pp. 649-659, 2010.

[10] P.V.A. Mohan, "Generation of OTA-C filter structures from active RC filter structures," *IEEE Trans on Circuits and Systems*, vol. 37, no.5, pp. 656-660, 1990.