Graph-based symbolic, symbolic noise and sensitivity analysis of CMOS analog integrated circuits

by

Santiago Rodríguez Chávez

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER IN SCIENCES WITH SPECIALITY IN ELECTRONICS

at

Instituto Nacional de Astrofísica, Óptica y Electrónica

Supervised by:
Dr. Esteban Tlelo Cuautle
INAOE

PhD Sheldon X.-D. Tan
University of California Riverside

July 2012
Tonantzintla, Puebla

A mis padres Belén y Rubén que siempre me han apoyado.

A mis hermanos.

A mi amada Pili.

Acknowledgements

My gratitude to all the professors involved in the Integrated Circuit Design program at INAOE for their effort and dedication in teaching new generations.

Special thanks to my advisor Dr. Esteban Tlelo Cuautle for guiding my through this work as well as providing valuable insight into the mechanics of research publication.

Thanks to my co-advisor Prof. Sheldon X.-D Tan from MSLAB at the University of California Riverside for providing me the chance to work at his lab. I hope this is just the first of a long series of research collaborations.

I would like to express my gratitude to the National Institute of Astrophysics, Optics and Electronics (INAOE) to give me the opportunity to have studied into an institution of academic excellence.

Finally, I want to acknowledge to the Consejo Nacional de Ciencia y Tecnologia (CONACyT), for the financial support during my master studies.

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

Abstract

Increasingly complex electronic devices are driving the chip circuit elements density towards the limit of the physically possible. The Metal Oxide Semiconductor by it's conveniently low cost of manufacture is the best option for the implementation of most electronic circuits. It is a very mature technology in terms of implementation. However, modeling of such devices is very complex and with more complete models comes increasing computation cost at simulation time. Model complexity increases as technology nodes decrease the minimum dimension. Many detrimental factors which were considered of second importance are now becoming even more notorious.

For instance, the Integrated Circuit designer now has to deal with low rail-to-rail voltages which make the use of some topologies impossible or has to deal with the fact that the behavior of a MOS transistor (MOST) deviates significantly from the hand equations (ie. quadratic model, variable depletion layer model) and interactions between MOST elements become even more complex due to the increased importance of secondary effects which play for and against each other. With the scaling down of supply voltages, signals in an IC are becoming of comparable magnitude to noise. Then there is the economic aspect which has to do with the design for manufacturing in order to increase the profit margin and/or decrease the cost to the final consumer.

The designer has to somehow get an insight of said interactions in order to cope

with them. Symbolic analysis, an area which sparked interest in the 1970's and 1980's lost it's momentum due to the difficulty in implementing true problem solving environments (PSE) [11]. In recent times however, the need to explore new paradigms along with the increased computing power lead to the integration of behavioral and numerical simulation in order to get the best of both worlds [5] [21].

The main contribution of this Thesis is the introduction of a graph-based technique for the solution of a determinant where all or at least one of its elements is a symbol. This technique is applied to compute symbolic transfer functions, symbolic noise expressions and symbolic sensitivities of analog integrated circuits composed of MOSFETs, whose behavior is modeled by using nullors. Several examples demonstrate the usefulness of the proposed graph-based technique and it is compared with the already known determinant decision diagram (DDD) method introduced one decade ago.

Resúmen

La creación the dispositivos electrónicos cada vez más complejos en tiempos recientes ha hecho que la densidad de integración de elementos circuitales en un chip tienda a alcanzar los límites de lo que es físicamente posible. Los dispositivos de Metal-Óxido-Semiconductor (MOS) por la conveniencia de su bajo costo de manufactura es la mejor opción para la implementación de la mayoría de los circuitos electrónicos de hoy en día. sta es una tecnología bastante madura en términos de implementación. Sin embargo, modelar dichos dispositivos es una tarea sumamente compleja y con modelos mas completos viene un inherente aumento en el costo computacional al momento de simular. La complejidad de los modelos aumenta conforme los nodos en la tecnología MOS disminuyen las dimensiones mínimas. Muchos efectos negativos que eran considerados de segundo órden se han vuelto más notorios.

Por ejemplo, el diseñador de circuitos integrados (IC por sus siglas en inglés) ahora tiene que lidiar con voltajes de riel-a-riel tan bajos que hacen imposible el uso de algunas topologías, o tiene que lidiar con el hecho de que el comportamiento de los transistores MOS (MOST) difieren significativamente de las ecuacionesde "a mano" (eg. modelo cuadrático, variable depletion layer) y las interacciones entre elementos MOST se vuelven mas complejas debido a que los efectos de segundo orden que juegan a favor y en contra son ahora más significativos. Con el escalamiento hacia

abajo de los voltajes de alimentación, las señales de interés en un IC son ahora de magnitud comparable al ruido en el mismo. Encima de todo esto está el aspecto económico que tiene que ver con el diseño para la manufactura en que se busca aumentar el márgen de ganancias y/o reducir el costo que el consumidor final tiene que pagar.

El diseñador entonces tiene que comprender dichas interacciones de alguna manera con tal de tomarlas en cuenta al momento de diseñar. El análisis simbólico, un área que ganó interés durante las décadas de 1970 y 1980, perdió el momento que obtuvo debido a la dificultad de implementar verdaderos Ambientes de solución de problemas (PSE, Problem Solving Environments, por sus siglas en inglés) [11]. En tiempos recientes, la necesidad de explorar nuevos paradigmas junto con el aumento y disponibilidad de poder de cómputo llevó a la integración de simuladores comportamentales-numéricos con el objetivo de obtener lo mejor de ambos mundos [5] [21].

La principal contribución de esta tesis es la introducción de una técnica basada en grafo para la solución de determinantes donde uno o más de sus elementos es un símbolo. Esta técnica es aplicada para calcular las funciones de transferencia simbólicas, expresiones de ruido simbólicas y sensitividades simbólicas en circuitos integrados analógicos compuestos con MOSFETs, cuyo comportamiento es modelado a través de nullores. Ejemplos para demostrar la utilidad de la técnica basada en grafo propuesta son presentados, además es comparado con una técnica ya conocida: determinant decision diagrams (DDD), que fué introducida hace una década.

Contents

Chapter 1

Introduction

1.1 Motivation

Nowadays portable electronic devices are ubiquitous. Personal, wearable computing and communication devices are incorporated into a single device. Such devices incorporate many functions which in the past were delegated to specialized separate devices such as geolocation devices (GPS), telephony, digital video and pictures, digital personal assitant, etc. On the other hand this increased pervasive power leads to the need for faster networks and higher amounts of storage, just to mention the two factors. So the market demands and increase in computing speed (eg. higher chip frequency), decrease in power consumption (low voltage, low-power), increasing integration of functions (eg. System on Chip, multi-core), etc. All this with a smaller or at least not bigger price tag to the final consumer.

The market drive for faster, more portable, low power electronic devices pushes the integrated circuit (IC) designer to move towards more complex designs increasing the number of elements and their density leading to technological wonders like the

now ubiquitous System on Chip (SoC). This is posible because of the ever changing chip technology moving from dimensions in the order of the hundreds towards the tens of nanometers and into new technologies like FinFET [37] [29] [23]. This comes at a cost to the IC designer which is now compelled to explore new techniques in order to get a better insight into the behavior of an integrated circuit. [12]. An immediate consequence of Metal Oxide Semiconductor (MOS) device scalling down is the increasing significance of second order effects (noise, channel lenght modulation, etc) [24]. It is possible to workaround such complications with optimization techniques and procedures. In the same manner, effects of high-order like mobility degradation due to transversal and longitudinal electrical fields should be considered in short-channel devices [18] [24].

A great deal of effort has been directed towards the analysis of these effects present in semiconductor devices and the related noise issue in the design of integrated circuits. The noise in semiconductors used to be considered of lower importance in the past, however, it becomes very significant as technology nodes approach the physical limit in deep submicron dimensions. As voltages are scaled down, the desired signal in a chip becomes smaller in magnitude and a prevoisly considered small noise perturbations is of comparable magnitude effectively increasing a circuit sensitivity to the various sources of noise [18] [37] [16].

The design of digital circuits by means of automatic tools is a very mature field. Existing tools are efficient, powerful and thus are widely used by designers. However, the analog counterpart in circuit design is still carried out mostly by hand. There is a lack of efficient and reliable analog computer aided design (CAD) tools which results in lenghty design times and costly design errors which lead to increasing number

of design iterations [32]. There is a real demand for industrial grade analog design automation (ADA) tools.

In the field of ADA an important task is the extraction and representation of the small-signal characteristics of interest in an exact (that is complete with no simplification) and approximate analytic form. These analytic expresions are of use in order to gain insight into the behaviour of the circuit. A huge research effort has been carried out in order to aid the designer in this field. The importance of this field is very well illustrated by the success of modem symbolic analyzers such as SCAD^3 [41], ISAAC [38] [14], ASAP [8] [9], SYNAP [39] [40], SAPEC [13], SSPICE [47], SCYMBAL [22], SCAPP [17], and GASCAP [19] for analog integrated circuits, as well as similar tools for symbolic Boolean analysis of digital circuits [3].

1.2 Symbolic Analysis

1.2.1 Definition of Symbolic analysis

Symbolic analysis is a formal technique quite useful to calculate the behavior or a characteristic of a circuit with its variables (dependent and independent) and with all or some of its circuit elements represented by symbols [13] [7]. This technique had a growing interest between 1960's and 1980's because of the increasing computing power and that many computer analysis techniques were proposed. A growing interest from the circuit design community started in late 1980's and this interest is seen in the success of modern symbolic analyzers introduced in [38] [14] [10] [9] [39] [40] [26] [47] [22] [17].

Knowledge of the behavior of a circuit is very important in its design process for

an analog integrated circuits (ICs) designer. For this reason, symbolic simulators become a very useful tool as they give as the result the analytic expression in a closed form of the circuit presenting the relationships between its parameters, this being and advantage over repetitive numerical simulations. Additionally symbolic analysis is useful when many numerical cancellations lead us to large roundoff error. However we can verify accurately the behavior of a circuit in a numerical simulation, this is specified only for a set of parameter values contrary to the symbolic simulation where the returned expression is valid even if the parameter values change (as long as the circuit topology remains the same).

Symbolic Analysis can be categorized in different approaches: Tree Enumeration methods, Signal Flow Graph Methods and Determinant based methods.Three Enumeration methods were proposed by Maxwell and Kirchhoff and first used for analysis of RCL networks represented as weighted undirected graphs. The advantage of this method is that the expressions are irreducible, however, for RLC-g_m networks the tree enumeration cannot be applied directly and tradeoffs between sign calculation and obtaining cancellation-free product terms are encountered [43]. Unlike Three Enumeration methods, Signal Flow Graphs are weighted directed graphs but the product terms obtained by this method are still not irreducible or cancellation-free. Determinant Based Methods are also not cancellation-free but it has been proven that topological methods (such as Tree Enumeration methods and Signal Flow Graph methods) don't offer any advantage over the later methods [43].

Any circuit transfer function can be obtained by applying Cramer's rule to the set of linear equations but solving the determinants in order to obtain the transfer function makes this an exponential space complexity method, in spite of that,

this drawback can be mitigated by applying recursivity and reusing data structure elements in order to reduce the memory footprint.

Chapter 2

Symbolic Formulation by Directed Graph

A very useful and studied tool for the formulation of symbolic expressions by solving a system of equations via Cramer's rule is the so called Determinant Decision Diagram (DDD). Thus Determinant Decision Diagram (DDD) based methods are classified as a determinant based method. DDDs are adapted from a data structure called Zero Suppressed Binary Decision Diagram (ZBDD) created in order to manage sparse subset systems. Two important observations on which determinant decision diagrams are based are that the admittance matrix representing a circuit is sparse and that a symbolic expression often shares many subexpressions [42]. In order to formulate the DDD, the determinant has to be expressed as a sum of products (SOP) which then are classified via a ZBDD in which every sum is a subset. Thus, DDD is an indirect method as it involves extra processing and the formulation of the full determinant beforehand.

In this chapter an approach for the solution of a system of equations based on

simple graph manipulations is introduced. The main difference stems from the fact that the graph structure in use is directly formulated from the matrix with no intermediate steps. Important observations on which this graph representation was developed are that the admittance matrix (in this case the nodal admittance matrix described in [36]) representing a circuit is sparse and that a symbolic expression often shares many subexpressions, characteristics that the DDD representation shares in principle. The determinant representation by applying our proposed graph-based technique is compact, unique and the complexity to obtain the symbolic expression depends on the size of the graph which in turn depends on the non-zero entries of the admittance matrix.

The code for all the functions was implemented in C and compiled in a RedHat Linux environment with the GNU C compiler gcc-4.2, Quad-Core Intel with 24GB RAM. ANSI-C compliance was considered of importance for migration purposes and universality, non-compliant functions were avoided.

2.1 Simple Case: Symbolic Determinant without node reuse

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. Every determinant has a unique representation, and is liable to symbolic manipulations. To understand how this approach works, lets us consider the following determinant [43]:

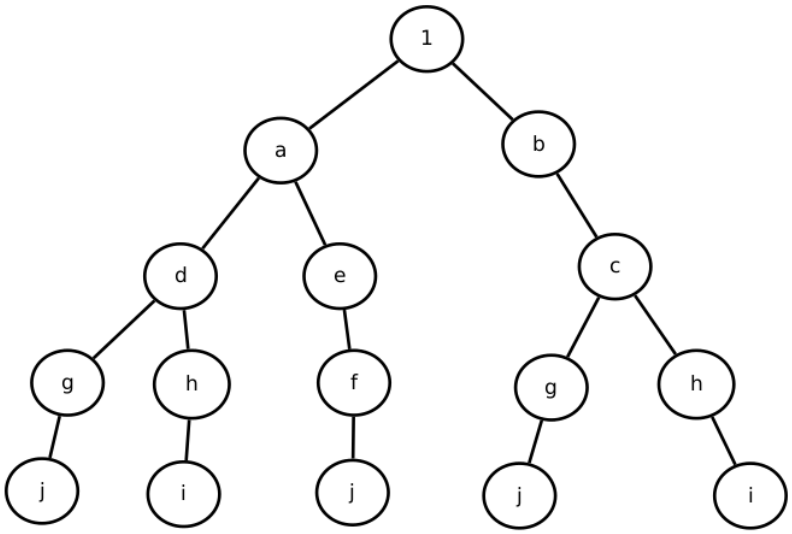


Figure 2.1: Graph representation of Equation (2.1)

$$det(M) = \begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj - adhi - aefj - bcgj + cbih \tag{2.1}$$

If the determinant's size is $n \times n$, we expect to have paths of $n + 1$ levels. So if there is a path that is not complete, i.e. that does not have $n + 1$ 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. This structure is built in a depth-first fashion. Every element of the graph structure corresponds to a non-zero entry in the admittance matrix. As a result, for this particular example we obtain the graph shown in Fig. 2.1, where all multiplications by zero were already omitted.

Path eliminations are performed by sending a prune signal if a zero is found inside the path. This prune signal is propagated all the way up until a summing point is

reached so the whole branch does not form part of the final structure and the path is terminated to zero instead. As one sees, this graph structure implemented is a tree in which the arithmetic operations are encoded in the depth of the tree node, that is, different depth implies multiplication while equal depth implies addition. This leads us to get the expression:

$$det(M) = a\left[d\left(gj - hi\right) + e\left(-fj\right)\right] + b\left[c\left(-gj + hi\right)\right] \tag{2.2}$$

A key point is related to the assignation of signs to each node in the expanded graph. They are established by applying the rule of signs from Cramer's rule. When applying graph methods to evaluate a determinant, not only one can obtain a factorized exact symbolic expression, but also derive all transfer relationships with respect to each node, and in a post processing step to each branch circuit variable. The graph representation shown on this section is compact for large class of analog circuits.

2.2 Advanced Case: Symbolic Determinant with node reuse

If the determinant is expressed as a SOP in graph form with no reuse of information the result is a graph with many repetitive terms (Fig. 2.1) corresponding to minors of the matrix that are repeated as was shown in the previous section. The smallest the matrix minor, the highest the repetition rate (if they are not zeroed out previously). In Fig. 2.1 the terms $(g \cdot j)$ and $(h \cdot i)$ are repeated twice. The information in each node is exactly the same, the only difference being the sign which in the end is trivial

as it can be easily computed.

The main idea in the algorithm presented in this work is that it is possible to reuse the information of those repeated nodes. For the previouse example it is possible to consider the products $(g \cdot j)$ and $(h \cdot i)$, and for correctness sake also the product $(f \cdot j)$ (which are the three non-zeroed product terms for the determinant of the 2x2 minors of M in Equation (2.1)) as independent subgraphs. So now we have five two-nodded subgraphs with vertex sets $V_1 = \{g, j\}$, $V_2 = \{h, i\}$, $V_3 = \{f, j\}$, $V_4 = \{g, j\}$ and $V_5 = \{h, i\}$. An important observation is that even though it may seem that edges $E_1 = \{g, j\}$ and $E_4 = \{g, j\}$ are equivalent as they convey the same operation which is the product $(g \cdot j)$ so the subgraphs $G_1 = \{V_1, E_1\}$ and $G_4 = \{V_4, E_4\}$ carry the same information and can be considered equivalent.

The reusing was possible because the information which made the graphs G_1 and G_4 different is somehow stripped from the subgraphs and obtained from somewhere else. It is possible to extend this idea even more in order to reuse the nodes in the last row by identifying the information which make nodes with same terms different from each other. The difference between the subgraphs $G_1 = G_4$ and $G_3 = \{V_3, E_3\}$ is the ancestor of node j. That is, for $G_1 = G_4$ it's node g so the subgraph represents the operation $g \cdot j$ whereas subgraph G_3 represents $f \cdot j$. Remember that each node is linked to a non-zero matrix entry[1]so there is a node for every $A_{row,col} \neq 0$, node f is in turn a representation of $A_{3,2}$ and node g is $A_{3,3}$. When visiting node j the question is *"Was the previous node f or j?"*. When selecting a pivot $A_{row,col}$ while formulating the determinant, row and column are removed, reformulating the previous question into *"Have we already visited a node from the same column as f?"* if the answer is affirmative then the previous node was g.

[1]It's sometimes useful to break a node representing $A_{row,col}$ term into two or more parallel

Extending this same procedure to the whole graph, the first obvious consequence is that there are no repeated nodes, in other words, for a matrix A with nz non-zero entries, there are nz nodes. So far information conveyed by relationships in the graph from Fig. 2.1 concerning to the sign of the adjugate matrix and the already visited columns has been stripped. The adjugate matrix sign can be computed as given by Equation (2.3) requiring to know row and column. Row is given already as the depth of a node making necessary to store the information pertaining to the column into the node.

$$\text{sign} = (-1)^{row+col} \tag{2.3}$$

The symbolic manipulations performed are agnostic of the origin of the matrix, that is, the only conditions are that the matrix is square. Each node in the graph structure corresponds to a matrix non-zero entry, which in turn encapsulates the summation of one or more circuit elements.

2.2.1 The Advanced Case in detail

Three different data structures are required. The first and most obvious is the node structure which contains the following fields:

- Node Name: A unique name for the given node. It is assigned as an index number.
- Terms: An array containing the index and sign of the element (mapping it to the element table described in section 3.1.1).

nodes.

- Column: The column of the non-zero entry in the matrix that this node belongs to.

- Descendants: An array of node pointers linking to the descendants of the current node in the graph structure.

The second data structure is a Graph type. It is useful to have a Graph structure with the fields:

- Graph Name: A unique name for the current Graph. It is possible to have many different Graphs. At least two Graphs are required at first: numerator and denominator.

- Matrix size: The size of the matrix. Only one dimension is required as the input is expected to be a square Matrix.

- Root node: The root of the Graph. In the simple AC analysis the root node is a trivial node with term value equal to 1 and column and row equal to zero. When multiple Graphs are constructed (ej. Factorization) the root node can be any of the nz nodes.

- Visited Columns When traversing a Graph to formulate the determinant the column of a visited node is appended to this array .

The third and final structure is a Matrix type. This structure stores not only the input Matrix but also the independent vector.

First off, $nz + 1$ nodes are created (nz = non-zero entries) and their respective structure fields filled in with the information read directly from the matrix. The

Row and Column field are useful to compute the sign rule when formulating the determinant as well as to determine which nodes are to be skipped.

The graph is built starting with a trivial node named 0 with term value of 1. Remember the multiplication is codified as depth in the graph. The different nodes are linked accordingly. The algorithm to build the graph structure for the representation of $|A|$ is shown as Algorithm 1.

Algorithm 1 buildGraph($A(i,j)$, *Ancestor*)

$m \leftarrow$ number of Columns of A
$n \leftarrow$ number of Rows of A
$D \leftarrow$ set of descendants of *Ancestor*
Ensure: $m > 0 \wedge n = m$
function BUILDGRAPH(A, *Ancestor*)
if $m > 1$ **then**
$prune = 1$
for $i = 1$ to m **do**
for all Columns j that $A_{i,j}) \neq 0$ **do**
if BUILDGRAPH($C_{ij}, A_{i,j}$) $\neq 0$ **then**
$D = D \cup A_{i,j}$
$prune = 0$
else
if $A_{i,j} \neq 0$ **then**
$prune = 0$
return *prune*

With the graph already built, the expression for the determinant is then formulated as in Algorithm 2.

The graph structure for the matrix in (2.4) is shown in Fig. 2.2.

Algorithm 2 Symbolic Determinant from Graph

1. Read the graph in a modified DFS fashion:
   (a) Keep track of which columns have been visited
   (b) Skip nodes from columns already visited
2. Symbolic expression is the product of a node times the summation of its visited descendants

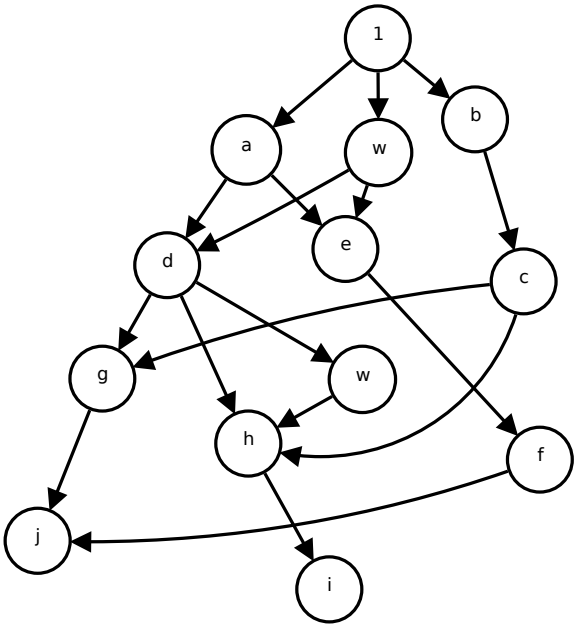


Figure 2.2: Determinant Graph

$$\begin{vmatrix} a & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{vmatrix} = adgj - adhi - aefj - bcgj + cbih \tag{2.4}$$

2.3 Symbol Factorization and Symbolic Derivative

2.3.1 Factorization of Symbol W

Symbol factorization is useful in order to ease numerical evaluation of the expression and in order to obtain the symbolic derivative with respect to a given symbol. Factorization takes place by following a simple algorithm (Algorithm 3). The result is the expression of the determinant as a polynomial represented by an array of sum of products with one entry for each power of symbol W with non-zero coefficient.

Algorithm 3 Graph to Polynomial

1. Expand the nodes of the graph containing the symbol W
2. Read the graph in DFS fashion and preserve only those routes from root to bottom with at least one occurrence of symbol W.
3. The number of occurrences of the symbol is the power of W for a given route.
4. Each route is then expressed as a sum of products replacing the symbol for a one.
5. The summation of all appended routes for each power of W forms the coefficients.

Suppose we have a matrix (2.5) and we want the polynomial form as powers of

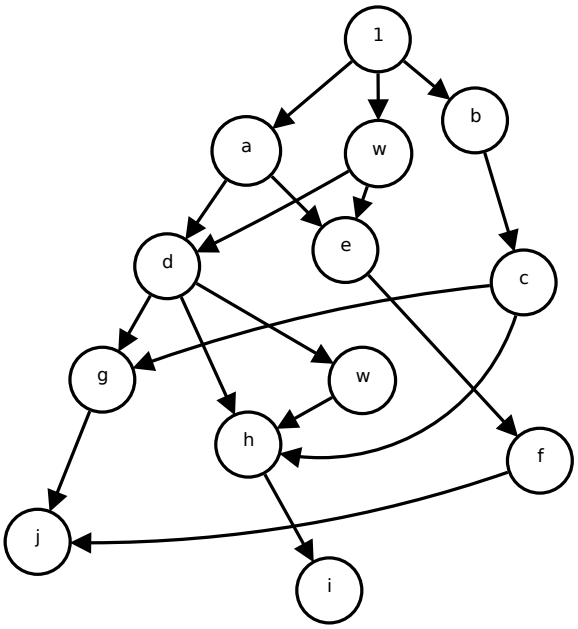


Figure 2.3: Expanded Graph

w. The resulting graph is shown in Fig. 2.3 and the polynomial expression is in (2.6).

$$M = \begin{bmatrix} a+w & b & 0 & 0 \\ c & d & e & 0 \\ 0 & f & g & h \\ 0 & 0 & i & j \end{bmatrix} \tag{2.5}$$

$$\begin{aligned} w^0 &= adgj - adhi - aefj - bcgj + bchi \\ w^1 &= adi + dgj - dhi - efj - bci \\ w^2 &= di \end{aligned} \tag{2.6}$$

2.3.2 Symbolic Derivative with respect to W

The derivative $\frac{\partial|\boldsymbol{A}|}{\partial W}$ is straightforward as symbol W in the expression $|\boldsymbol{A}|$ has already been factorized. The graph expansion can be performed in the original structure generated when obtaining the determinant thus reducing the memory footprint. Expansions for more than one symbol are possible.

2.3.3 Symbolic Mixed Derivative

It is possible to get a mixed derivative of the form $\frac{\partial^2 f}{\partial W_1 \partial W_2}$ for symbols W_1 and W_2. For this purpose we make use of the resulting symbolic expression obtained previously in Section 2.3.2. Such expression is in the form of a sum of products, which is further separated in n coefficients (powers of W_1 with coefficient 0 are omitted) of W_1. Each coefficient is expressed as a graph, thus obtaining n graphs. The derivative procedure in Section 2.3.2 is repeated for all n graphs individually now for the symbol W_2. This is useful for some optimization problems where the Hessian matrix needs to be computed [2].

Chapter 3

Symbolic Analysis with MOSFET elements

3.1 Implementation of the Graph-Based Approach

The proposed graph-based approach for the solution of a system of equations starts off with a SPICE netlist as input. The elements implemented are R, C, L, V, I, E, G and M, being resistor, capacitor, inductor, independent voltage source, independent current source, voltage controlled voltage source, voltage controlled current source and MOSFET. Both independent voltage and current sources can be DC, AC or both.

The netlist is to be used as a way to input the circuit topology along with circuit elements and values. If the numerical evaluation of the resulting symbolic transfer functions generated is required, the small signal values for the parasitic elements and the operating point conditions are read from the output listing in $HSPICE^{TM}$ format. This numeric values are of use when evaluating the transfer function to

verify correctness as well a way to rank the sensitivity. The flow of the tool is shown in Fig. 3.1

The derivative is computed in an alternative module which is bypassed when the only output required is the actual transfer function. The algorithm to implement this symbolic derivative accommodates for consecutive derivatives with respect to different variables (symbols). The flow of this module is presented in Fig. 3.2.

3.1.1 Parsing the netlist

The first step is to parse the netlist and build suitable data structures for each group of elements. The symbol given to every device from this point onwards is the same as its name. To keep consistency, the symbol name is taken exactly as specified in the netlist (ex. R_name, C_name, M_name, etc) input is not case sensitive and is parsed as lowercase. Elements are first grouped into one of four different tables: conductances, independent sources, controlled sources and Mosfet Table 3.1.

Table 3.1: Elements Tables

Table Type	Fields
Conductances	Name, Node 1, Node 2, Value
Iindependent Sources	Name, Node 1, Node 2, DC, AC
Controlled Sources	Name, Node 1, Node 2, Node 3, Node 4, Gain
MOSFET	Name, Drain, Gate, Source, Bulk, Width, Length, ModelName

Values for elements and sources are considered strings of characters so the tool is agnostic of whether they are actually numbers or functions until the numerical evaluation of the output is performed. The symbol name is always the element identifier and it's name exactly as it appears in the netlist. The network relationships

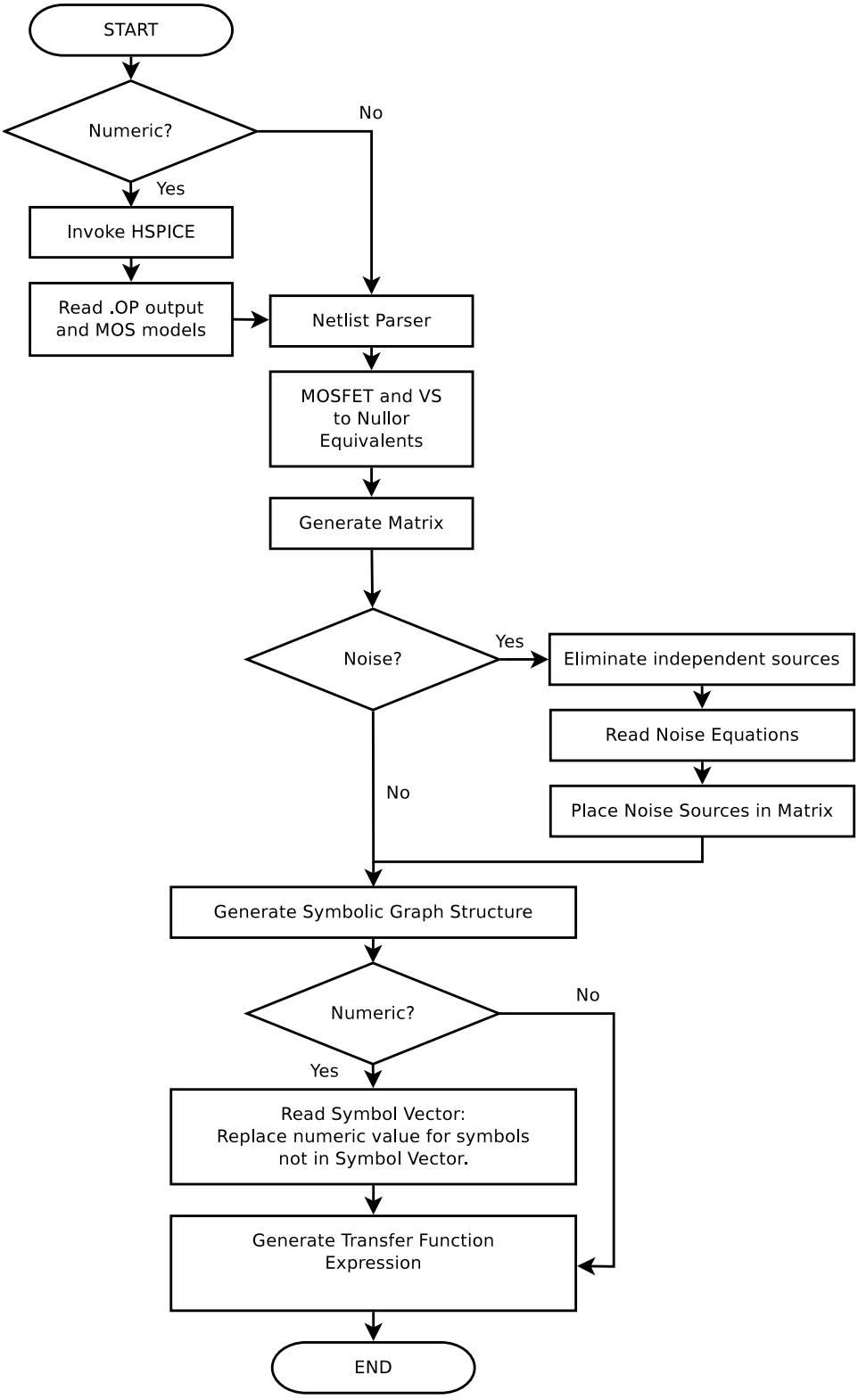


Figure 3.1: Symbolic Tool Flow

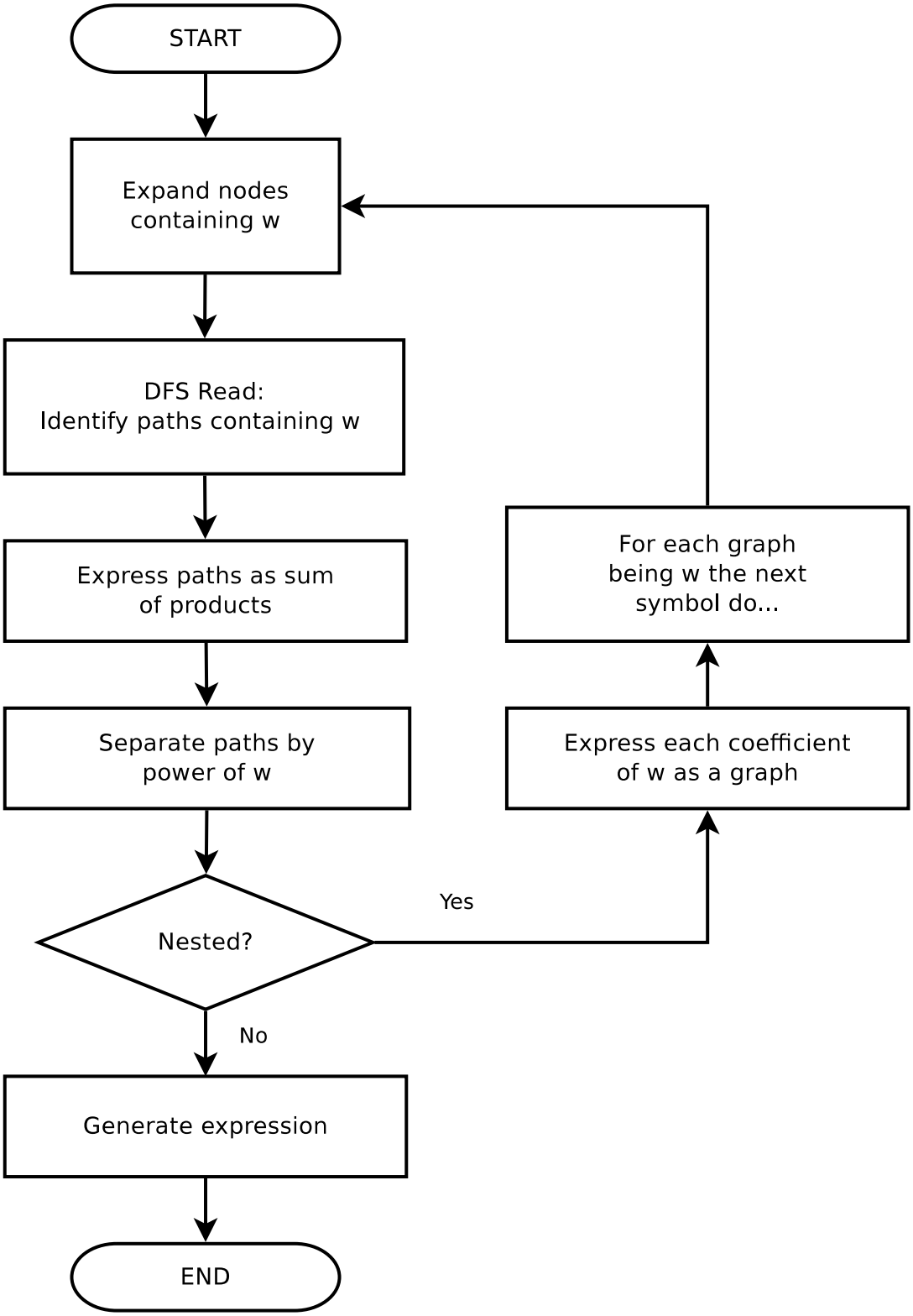


Figure 3.2: Module for Symbolic Derivative

of the different elements are codified in their node connections. Because it is a lot easier to treat nodes as integer indexes, a mapping is created first by building an array with all the nodes as they appear in the netlist and any duplicate occurrences are removed.

The tool is intended to perform AC analysis so it is of no use to keep independent sources with only DC value, as they are only valid in a DC or transient context. When removing voltage DC sources their nodes are collapsed, while current DC sources nodes are left floating. Collapsing two nodes in a circuit is equivalent to assigning to both nodes the same mapping and so is done in this step. If the numerical evaluation of the resulting transfer function is to be performed, the output listing (.lis file for $HSPICE^{TM}$) is read and the values for the operating point are parsed and stored in an array with the same index correspondence as the element symbol array so that the mapping between the symbols and their numerical values is direct.

Up to this point the elements are separated in groups according to their type, nodes are collected in a single array and are mapped to indexes to facilitate their treatment, DC sources are removed.

3.1.2 Small Signal Models and NullOr Equivalents

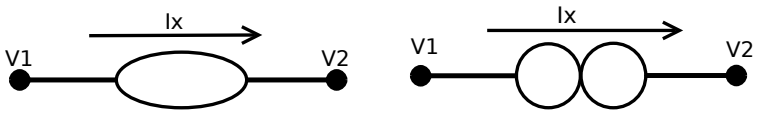
In the present implementation, active elements are substituted by their controlled source based small signal model. In turn, controlled sources are modeled with combinations of norator and nullator in order to make use of the extensive studies of NullOr based circuits. The Nullator and Norator are abstract elements with ideal characteristics [7]. The voltage across the nullator terminals is zero and does not allow current to flow through it. Even though it has some properties of an open

circuit it doesn't have an immittance or a scattering representation.

$$V_1 = V_2 = \text{arbitrary}, I_x = 0 \tag{3.1}$$

For the norator, the voltage between its terminals is arbitrary and an arbitrary current can flow through it. The norator also shows some properties of an open circuit. The circuit representations of the nullator and norator are shown in Fig. 3.3a.

$$V_1 \neq V_2 = \text{arbitrary}, I_x = \text{arbitrary} \tag{3.2}$$



(a) Nullator. (b) Norator.

Figure 3.3: Pathological element symbol for a) Nullator and b) Norator.

One can combine these elements in order to obtain a new one called Nullor where,when it is modeled as a two-port element, its first port is the Nullator and the second is the Norator, that is why the Norator and Nullator receive the name of Nul-

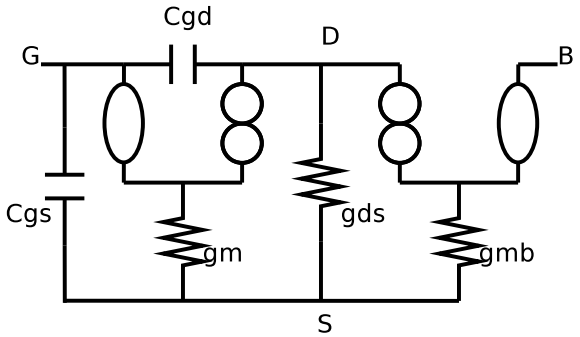


Figure 3.4: Nullor representation of the MOSFET

lor elements. The Nullator was first introduced by Carlin [4] . This simple element has proven its usefulness in areas like symbolic analysis. One of the main advantages of the Nullor elements is that nodal analysis (NA) of an active circuit is simple and also they can model active circuits independently of the particular realization of the active devices [7] [46]. For the MOSFET, the Nullor equivalent including the parasitic gate-source and gate-drain parasitic capacitances used in the sensitivity analysis is shown in figure 3.4, therefore for an operational transconductance amplifier (OTA) its Nullor equivalent would be as shown in Fig. 3.5

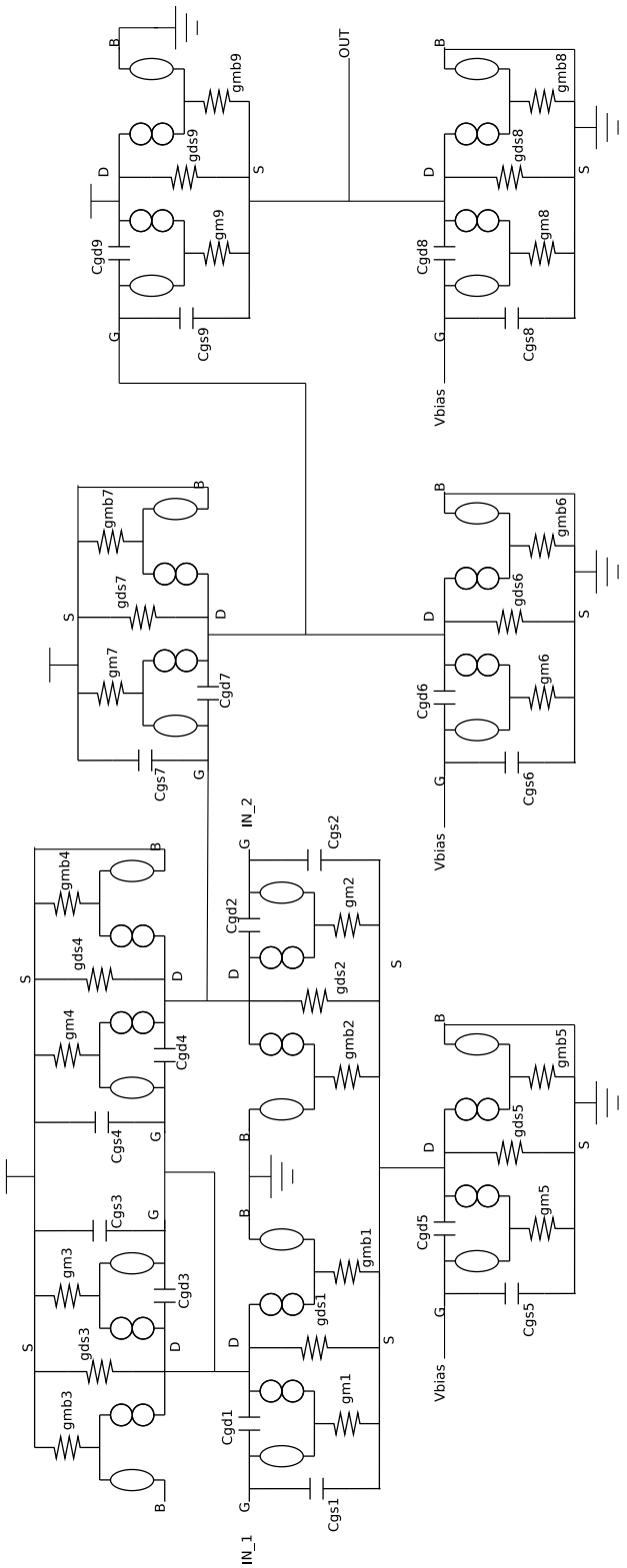


Figure 3.5: Nullor representation of a three stage OTA

The guidelines for obtaining the nodal admittance matrix by applying nodal analysis is summarized below for convenience and is explained in detail in [36], [34].

Two sets of pairs of nodes are formed, one for ROWs and one for COLs. These two sets are then used to form the admittance matrix by performing the Cartesian product of every subset. In the ROW group a subset is formed for every node with no Norator connected to it, and a different subset for every group of nodes connected by floating Norators. In the COL group a subset is formed for every node with no Nullator connected to it, and a different subset for every group of nodes connected by floating Nullators. Two groups of admittances are formed, the first (group A) containing a subset for every node listing all the admittances connected to it, the other (group B) listing floating admittances with the corresponding pair of nodes. If a node is present in a subset in ROWs and a subset in COLs then the corresponding subset of admittances (from group A) is summed at the matrix position (ROW index, COL index). If a pair of nodes is present, one in a subset of ROWs and the other in a subset of COLs, the corresponding admittance (from group B) is summed with negative sign at the matrix position (ROW index, COL index).

Since the method applied is symbolic NA, the independent voltage source is transformed to a current source equivalent circuit. In order to simulate a differential input in circuits like the differential pair and the three stages operational transconductance amplifier (OTA) a voltage controlled voltage source which is converted also to a nullor equivalent is used. The nullor representation of both elements traditionally non-NA compatibles are shown in Fig. 3.6.

In a previous section it was explained why symbolic modeling is intended to give an insight into certain behaviors and tendencies of the circuit in question. With this

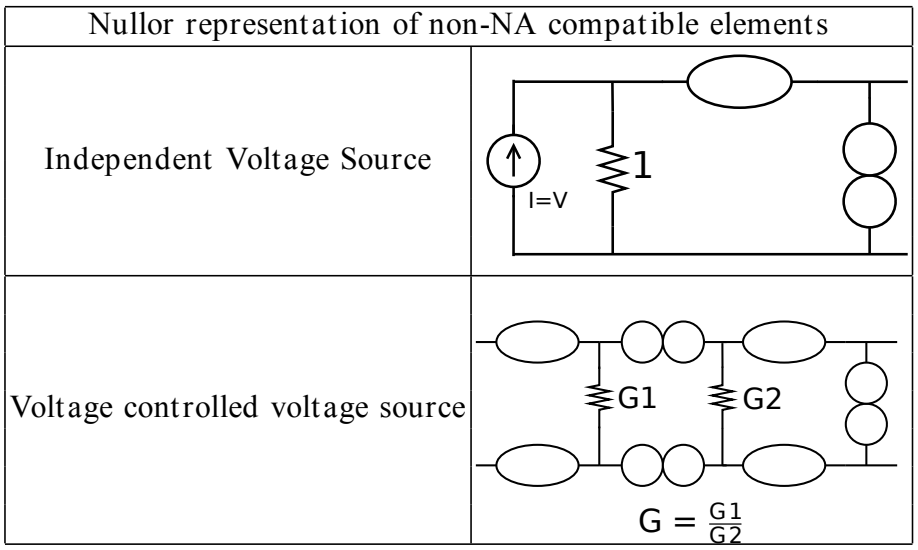


Figure 3.6: Nullor representation of non-NA compatible elements

in mind, it is now evident that the more complex the small signal model of a device the more accurate is the resulting simulation but at the cost of increasing machine operation time. A compromise can be reached where the qualitative behavior is roughly preserved at the expense of numerical accuracy. Implemented are three levels of parasitic elements for the small signal model for the Mosfet:

- Level 0 has no parasitic elements and models only the voltage controlled current source with gate-source as the controlling branch voltage and transconductance *gm*.
- Level 1 accounts for level 0 plus *Cgs, Cgd* and *Gds*.
- Level 2 accounts for level 1 plus the voltage controlled current source whose controlling branch voltage is bulk-source with transconductance *gmb*.

From this first stage the elements are classified as *NA compatible* or *NA incompatible* according to their small signal models. Incompatible elements are then treated

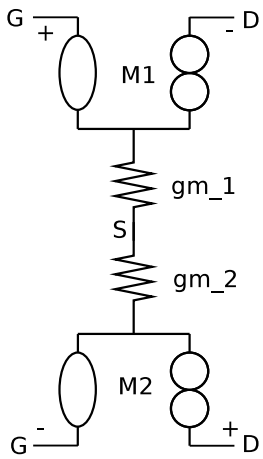


Figure 3.7: Nullor Equivalent of the MOSFET Differential Pair

as their respective Nullator/Norator (Nullor) equivalents as shown before in Fig. 3.6.

There are two reasons why elements can be incompatible. Firstly if the dependent variable of the function for a given element is voltage, as is the case for voltage sources and secondly if the independent variable for the element function is current. The reason behind this, as the reader may already be aware, is that in nodal analysis the unknown vector is composed of voltages (node voltages) while the independent vector is formed by current sources. Then, the need arises to apply a manipulation such that those conditions are preserved while maintaining a physical equivalence.

On the other hand, a fully differential OTA can be modeled as a single VCCS, which in turn can be viewed as a pair of MOSFET transistors in differential input configuration Fig. 3.7.

The ideal conditions for each of the two MOSFETs are kept: infinite input impedance (zero branch current in input Nullator), zero output impedance (arbitrary voltage) and ideal gm. Parasitic elements can then be arranged around this basic block.

As explained before a set of Nullator-Norator equivalents are implemented to account for the intrinsic voltage controlled current source of the Mosfet small signal model Fig. 3.4, voltage sources and controlled sources Fig. 3.6.

The basic building blocks for the equivalents are the Voltage Follower 3.8a and the Current Follower 3.8b

(a) Voltage Follower (b) Current Follower

Figure 3.8: Nullor based (a) Voltage Follower and (b) Current Follower

When replacing a certain element with it's NullOr equivalent there are three basic operations that have to be performed:

1. Add element to the proper array (nullator, norator, conductance or independent source)
2. Add extra nodes if required (MOSFET, Voltage Source, etc.)
3. Assign unique name to element and the extra nodes (as required)

Adding a new element is as easy as appending the new entry to the corresponding structure (conductance, independent source, nullator or norator) and updating the number of elements for the given structure. As a list of the nodes originally read from the netlist is kept, it is easier to keep track of the nodes added afterwards if any new node is appended at the end of the structure containing the numerical mapping. When adding an element it is useful if the name is a composition of the name of the

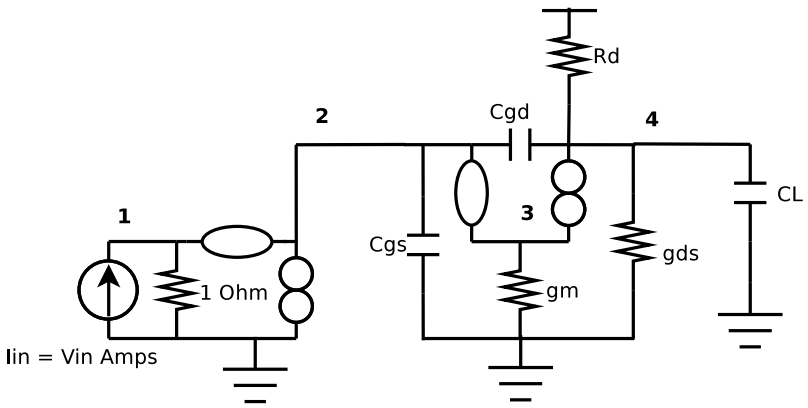


Figure 3.9: Common Source Small Signal Equivalent

original element it is bound to. For instance, if *CGS* is included into the model for the MOSFET named M_1, the name of the new parasitic capacitor can be *cgs_1*. In this way it becomes easier to tell to which netlist element a given symbol belongs to.

Now that there is a mapping of unique node elements, it is possible to begin replacing elements for their corresponding NullOr equivalents. For the case of the MOSFET the small signal model conformed with the selected parasitic elements and the corresponding NullOr is replaced in the circuit netlist.

3.1.3 Matrix equations formulation

Now that we know the node mappings we can create an adjacency matrix for each of the elements. From this adjacency matrix the NA formulation is performed by following the method in [36] which is reproduced here in a short form for convenience (Algorithm 4).

This procedure is shown for a simple common source circuit as the one shown in Fig. 3.9.

The formulated groups are shown in Table 3.2. The resulting matrix is a 2 x 2

Algorithm 4 NA Formulation with NullOr elements

1. Two groups of nodes are formulated, one named ROW and another named COL. The admittance matrix is formulated by performing the Cartesian product of these two groups.
2. Formulate ROW group of nodes:
   - Add a subset for each node with no Norator connected to it.
   - Add a subset for every chain of nodes connected by floating Norators.
3. Formulate COL group of nodes:
   - Add a subset for each node with no Nullator connected to it.
   - Add a subset for every chain of nodes connected by floating Nullators.
4. Group A: Admittances seen in a node.
   - A set for each node containing all admittances conected to it.
5. Group B: Floating Admittances.
   - A set for each floating admittance containing its pair of nodes.
6. Formulate Matrix
   - If a node is present in a subset in ROWs and a subset in COLs then the corresponding subset of admittances (from group A) is summed at the matrix position (ROW index, COL index).
   - If a pair of nodes is present, one in a subset of ROWs and the other in a subset of COLs, the corresponding admittance (from group B) is summed with negative sign at the matrix position (ROW index, COL index).

Table 3.2: NA Formulation

ROWS	COLS	FLOATING	SEEN BY NODE
(1)	(1,2,3)	(2,4) : sC_{gd}	$1 : 1Ohm$
(3,4)	(4)		$2 : sC_{gs}, sC_{gd}$
			$3 : gm$
			$4 : sC_{gd}, g_{ds}, \frac{1}{R_d}, sC_L$

system as shown in Equation 3.3.

$$\begin{array}{c} \\ (1) \\ (3,4) \end{array} \begin{array}{c} \begin{array}{cc} (1,2,3) & (4) \end{array} \\ \begin{bmatrix} 1 & 0 \\ gm - sC_{gd} & -(sC_{gd} + g_{ds} + \frac{1}{R_d} + sC_L) \end{bmatrix} \end{array} \begin{bmatrix} v_{(1,2,3)} \\ v_{(4)} \end{bmatrix} = \begin{bmatrix} I_{in} \\ 0 \end{bmatrix} \tag{3.3}$$

The solution for this system of equations becomes an application for the symbolic tool presented in this work. The system of equations is solved by using Cramer's rule by computing $n+1$ determinants for $[v_1, v_2, \cdots, v_n]^T$ node voltages in order to completely define the circuit. For the system of equations $\boldsymbol{A}x = b$ where $\boldsymbol{A}$ is n x n matrix, the solution by Cramer's Rule is given as $x_i = \frac{|\boldsymbol{b} \rightarrow \boldsymbol{A}_i|}{|\boldsymbol{A}|}$.

3.1.4 Experiments

For the experimental verification the symbolic transfer function of four CMOS and one BJT (small signal model with Rpi, Cpi, Cmu and Gm VCCS) circuits was computed and time metrics were taken for matrix equations formulation, denominator,

numerator and transfer function computation. The test cases are the differential pair and common source shown in Fig. 4.2, the three stages uncompensated OTA in Fig. 3.5 [15], recycling folded cascode OTA [31] and 741 OPAMP respectively as shown in Table 3.3.

Table 3.3: Symbolic Formulation and Numerical Evaluation of D(s), N(s) and H(s)

Circuit Features			Computer Time (seg)			
Circuit	Elements	Nodes	Equations	D(s)	N(s)	H(s)
Differential Pair	35	26	1.1235	0.122	0.1464	1.4895
RFC OTA [15]	106	56	1.6603	0.201	0.1869	2.2633
LV Amp	33	18	2.35	0.058	0.0464	2.4544
Common Source	8	6	0.8581	0.041	0.0205	0.9811
741	112	77	0.5123	1.37	0.822	2.7043

The test circuit for the UA741 is a Small-Signal model with R, C and VCCS elements. This same circuit was tested with the SCAD3 [43] which does not support reading MOST from netlist, in addition SCAD3 documentation states that the evaluation has to be performed twice for some obscure reason. The running and evaluation time reported by SCAD3 is of 2.97s while our tool takes 2.7043s.

Numerical evaluation of the resulting symbolic expression is performed in order to provide a comparison between a well known and mature technology (HSpice^{TM}), a previously developed symbolic tool [42] and our tool. The test circuit is the same Differential Pair from Fig. 4.2b where the input to SCAD3 has been converted to VCCS equivalents. The AC analysis output is shown in Fig. 3.10.

Numerical evaluation of the resulting expressions for the RFC OTA [15] are shown in Fig. 3.11.

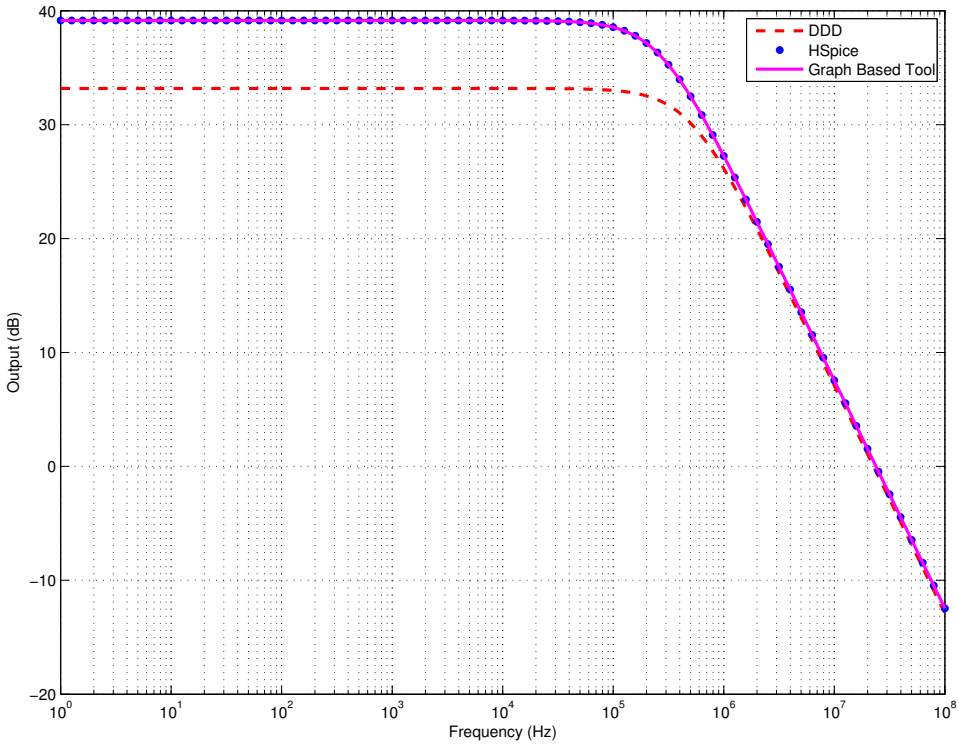


Figure 3.10: H(s) comparison between DDD (SCAD3) [42] (dashed), HSpice^{TM} (dots) and Graph Based Tool (solid) for the differential pair topology.

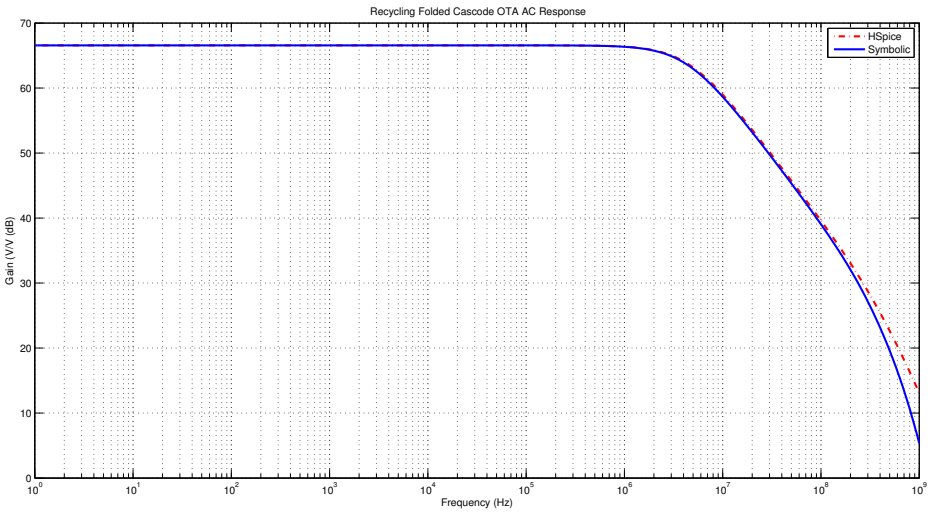


Figure 3.11: H(s) comparison between HSpice^{TM} (dashed) and Graph Based Tool (solid) for the RFC OTA [15] topology.

Chapter 4

Symbolic Noise Analysis

Noise is an effect present in passive elements as well as in active ones. It can internal or external, random or repetitive. Usually it is modeled by a current source which can be valid only in a definite band of frequencies [48]. In a MOST there are three main sources of noise: thermal, flicker and shot noise [24] [12] [45]. Importance of noise in the drain current of a MOST is of great importance for analog and RF circuits [33].

The traditional approach to noise analysis is with the use of numerical simulators given the high speed of operation. However, a numerical simulator does not show exactly which element in the circuit is noisier.

Noise is defined as any signal whose amplitude is random in function of the time. The amplitude of such signal at a given instant of time cannot be predicted although past values are known. Noise is characterized statistically rather than as a deterministic quantity. Noise in electronic circuits is caused by tiny fluctuations in current or voltage. The noise magnitude sets a lower limit on the signal that a given system can detect and effectively react to. Noise can be reduced and shaped but not

eliminated since it is due to fundamental mechanics in physical implementations of electronic circuits.

Chapter 2 of [35] gives a very thorough treatment of noise in electronic circuits.

4.1 Noise Sources and equations

To perform noise analysis, a thermal noise current source is attached in parallel to every resistance (Fig. 4.1a) (capacitors and inductors are considered noiseless) and a noise current source accounting for both thermal and flicker noise connected between drain and source for every MOSFET (Fig. 4.1b). Each of these noise sources has a value represented by a symbol which is substituted by the corresponding symbolic noise equation [1] shown on Table 4.1 just after the transfer function is formulated.

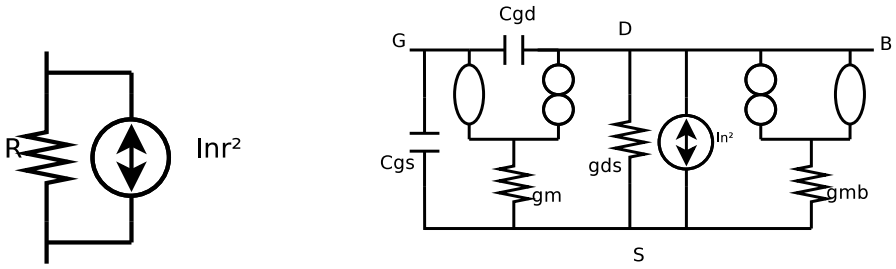


Figure 4.1: Noise Sources in (a) Resistor and (b) MOSFET

In order to solve the system for a given node as output, at least two determinants are calculated: one for the original admittance matrix in order to obtain the denominator and the other for the admittance matrix with the independent sources vector substituting the column of the desired output node to obtain the numerator. Additional determinants for the denominator must be computed if superposition of independent sources has to be performed. Symbolic determinants are formulated

Table 4.1: Noise Equations

NOISE MODEL	THERMAL NOISE CURRENT SOURCE	FLICKER NOISE CURRENT SOURCE
Resistor	$\frac{4}{3}kT\frac{1}{R}$	–
NLEV 0	$\frac{8}{3}kTg_m$	$\frac{KF \cdot ID^{AF}}{(C_{OX} \cdot Leff^2 f)}$
NLEV 1	$\frac{8}{3}kTg_m$	$\frac{KF \cdot ID^{AF}}{(C_{OX} \cdot Leff^2 \cdot Weff^2 f)}$
NLEV 2	$\frac{8}{3}kTg_m$	$\frac{KF \cdot g_m^2}{(C_{OX} \cdot Leff^2 \cdot Weff^2 \cdot f^{AF})}$

by using graph structures as described in Section 2 where a symbol is generated for every element of the Nullor equivalent circuit which conveniently contains only Norators, Nullators, impedances and current sources.

4.2 Numerical evaluation and validation

A final step consists of evaluating both $s = wj$ and f (frequency in flicker noise Equations 4.1) symbols in order to plot the output noise versus frequency, the output noise table reported in the output listing of the $HSpice^{TM}$ simulation is read and stored in vectors and then a superimposition of both plots is shown in order to compare results.

Three amplifier circuits are [28] [6] evaluated: common source, differential pair and an uncompensated three stages amplifier.

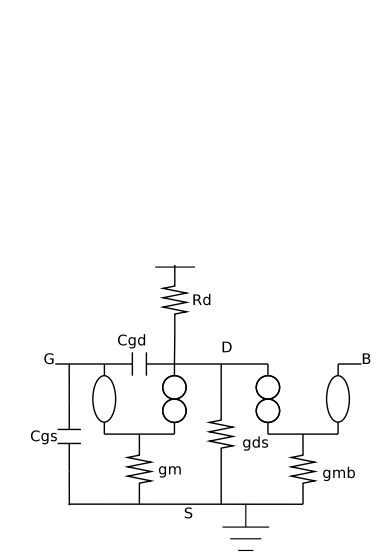
Noise sources are added as explained before: one current noise source accounting for both thermal and flicker noise for each transistor and a current noise source for every resistance.

The symbolic voltage noise output expression for the amplifier in Fig. 4.2a is formulated (4.1) for NLEV=0. It is evident that the automatic results provided

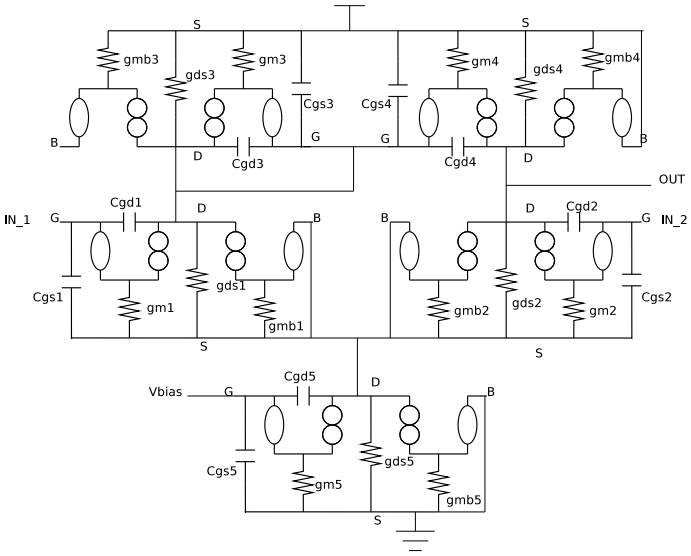
by the tool coincide with hand calculation for the output noise. Remember that for NLEV 0 the term gm^2 is not present and instead ${I_D}^{AF}$ is used.

$$V_{n,out}^2 = \frac{\frac{8}{3}kt(gds + gm + gmb) + \frac{4kt}{r_d} + \frac{{I_D}^{AF} \cdot KF \cdot TOX}{Leff^2 \cdot EOX \cdot f}}{(gds + s \cdot cgd + 1/r_D)^2} \tag{4.1}$$

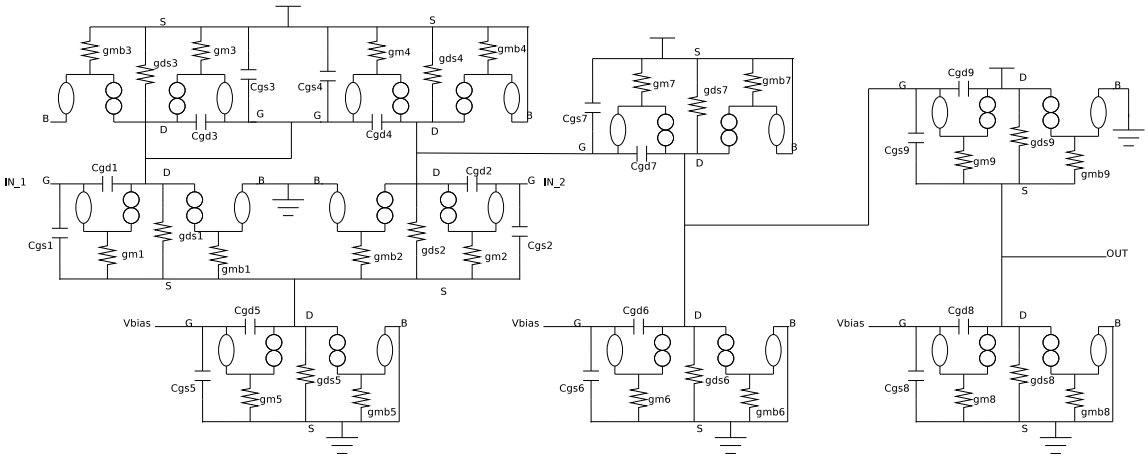
The symbolic expression for the common source amplifier is then evaluated and plotted against the $HSpice^{TM}$ results in Fig.4.3 for NLEV 0 and in Fig.4.4 and Fig.4.5 for NLEV 1 and 2 respectively. In Figs.4.6, 4.7, 4.8 the responses for the differential pair and Figs.4.9, 4.10, 4.11 the responses for the three stage amplifier are plotted for NLEV 0, 1 and 2.



(a) Common Source.



(b) Differential Pair.



(c) Three stages uncompensated amplifier.

Figure 4.2: Nullor equivalents of the amplifier circuits.

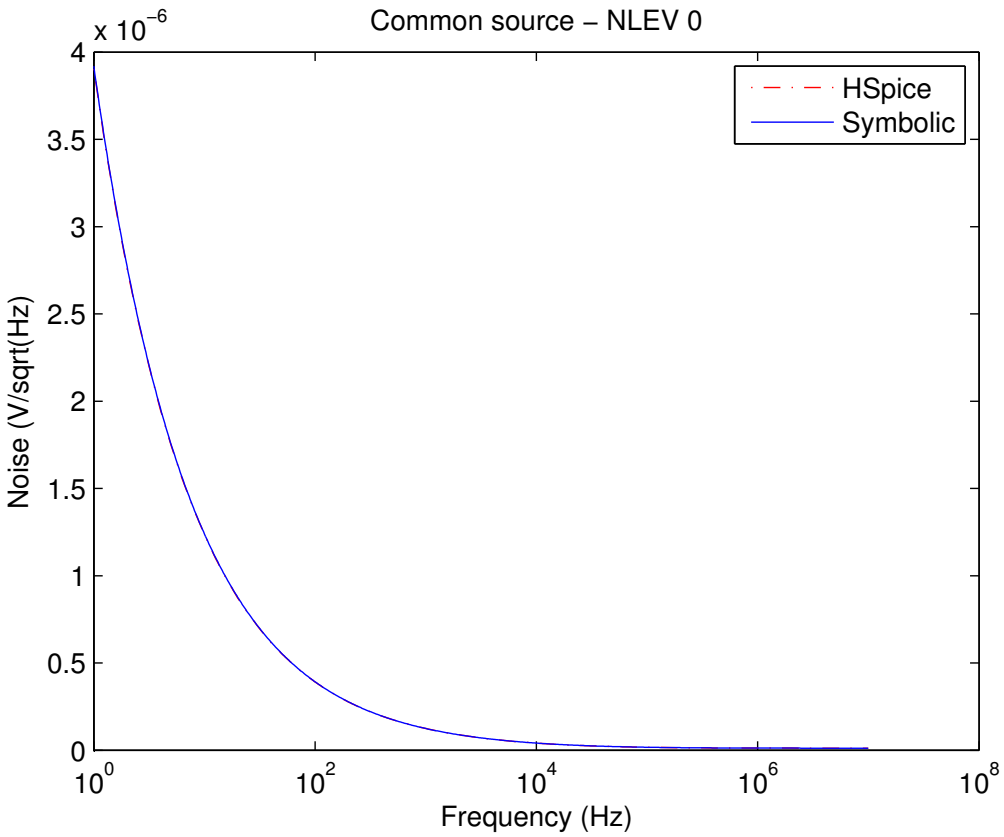


Figure 4.3: Common Source Output response with Nlev 0 Noise Equations.

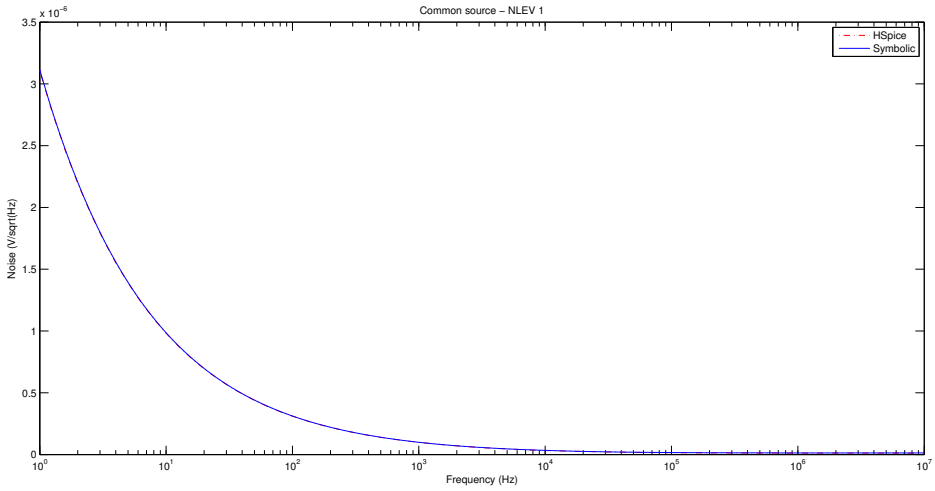


Figure 4.4: Common Source Output response with Nlev 1 Noise Equations.

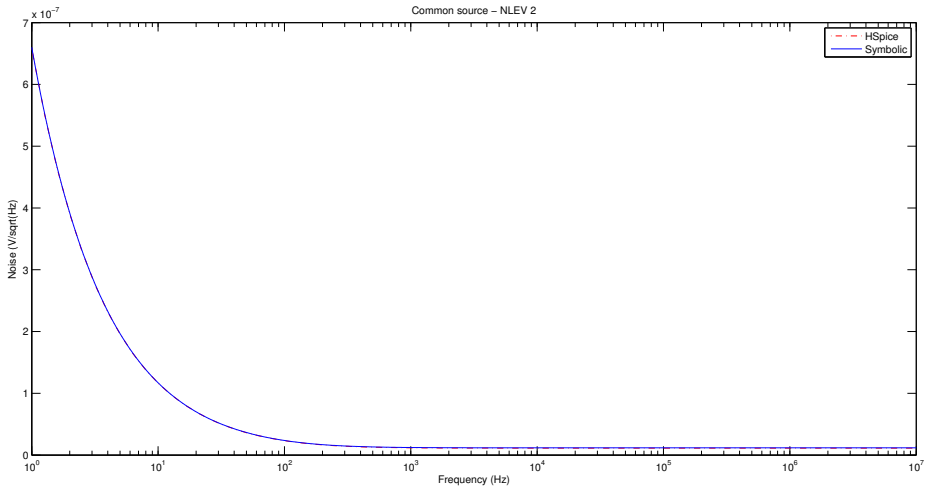


Figure 4.5: Common Source Output response with Nlev 2 Noise Equations.

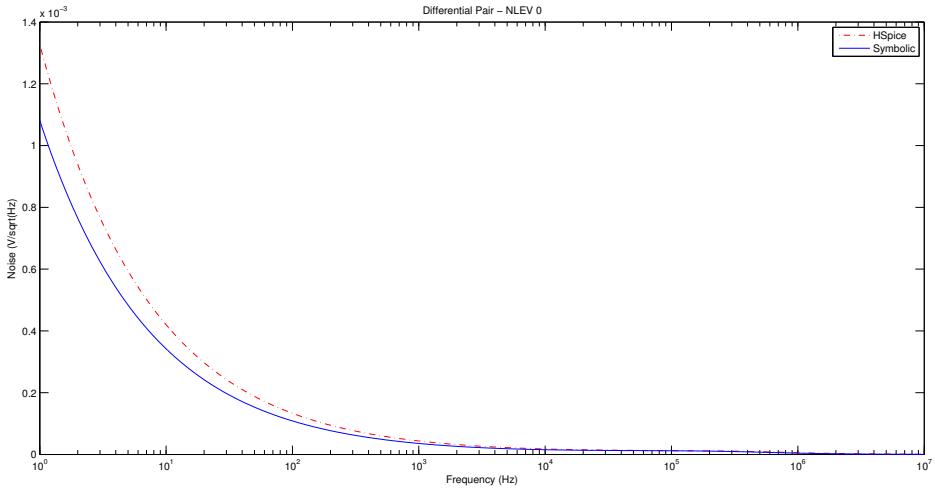


Figure 4.6: Differential Pair output response with Nlev 0 Noise Equations.

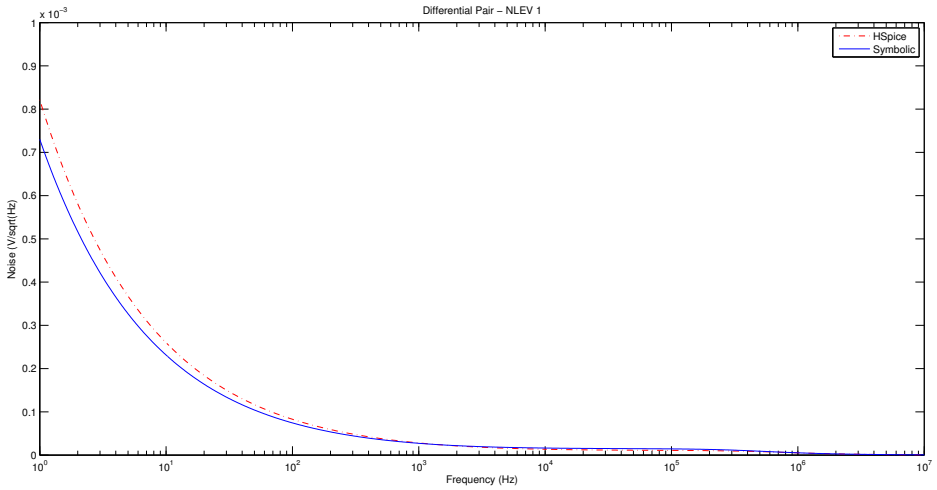


Figure 4.7: Differential Pair output response with Nlev 1 Noise Equations.

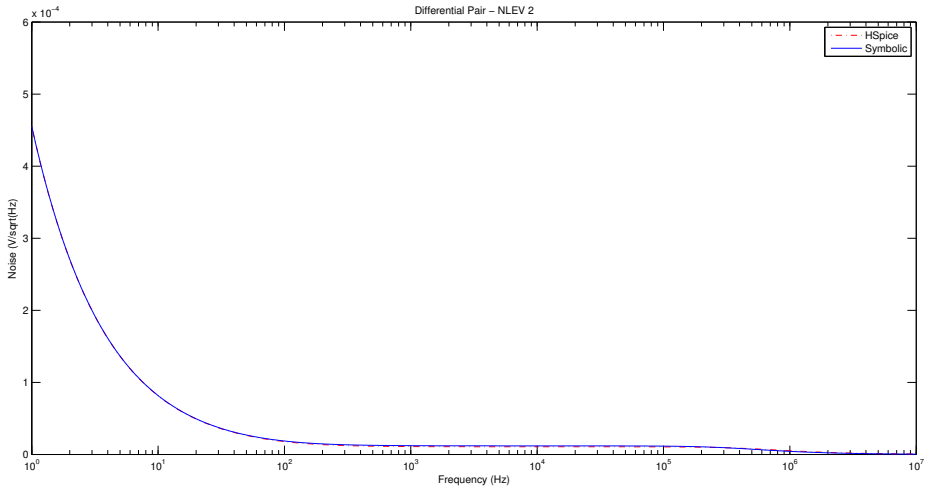


Figure 4.8: Differential Pair output response with Nlev 2 Noise Equations.

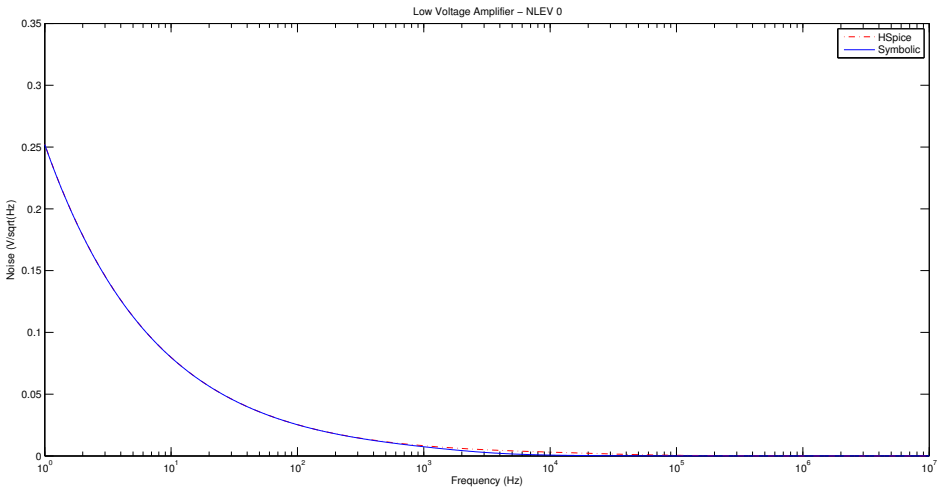


Figure 4.9: Three stage amplifier output response with Nlev 0 Noise Equations.

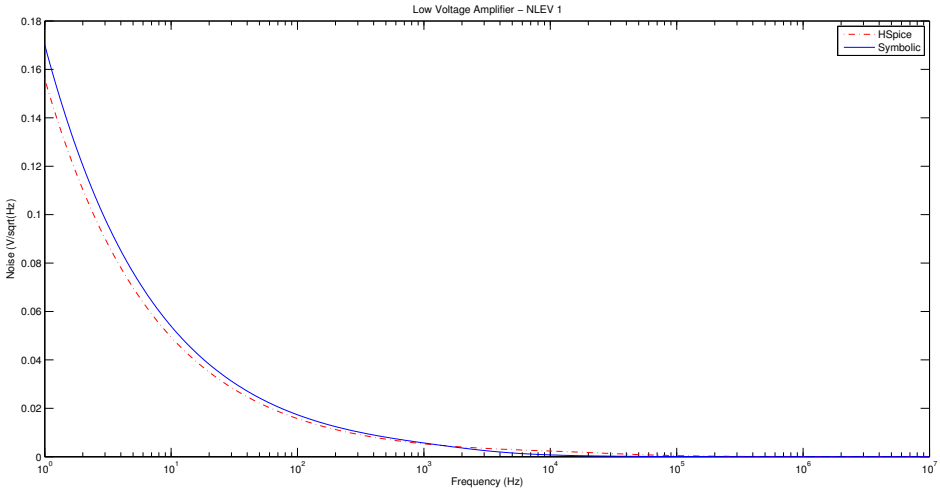


Figure 4.10: Three stage amplifier output response with Nlev 1 Noise Equations.

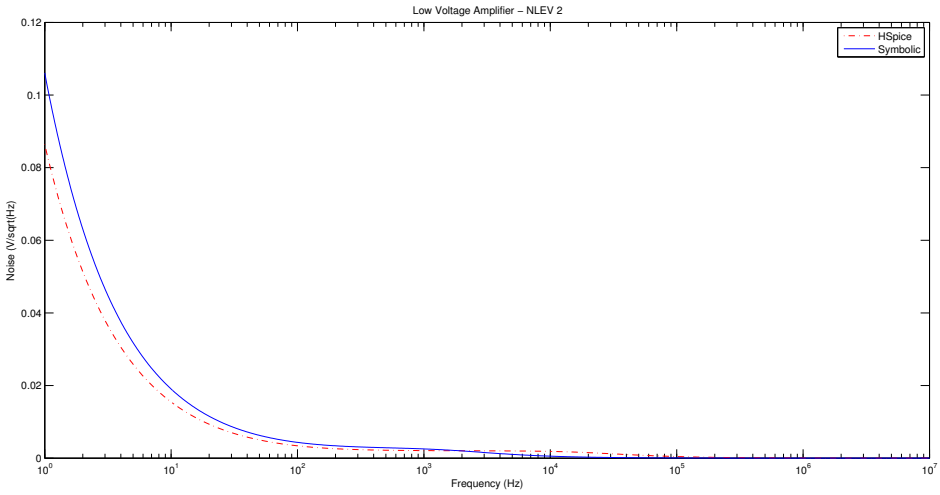


Figure 4.11: Three stage amplifier output response with Nlev 2 Noise Equations.

Chapter 5

Simbolic Sensitivity Analysis

Sensitivity analysis is very important in IC design since it helps us to optimize the behavior of a given circuit, by showing us which components of the entire system are more sensitive [30]. Also 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. Moreover, it is worthy mentioning that the advantage of our proposed graph-based technique is extended for the symbolic sensitivity analysis of analog ICs. We highlight that it is easier to compute sensitivities of any analytical expression with respect to one or many parameters, provided that a symbolic transfer function exist.

5.1 Sensitivity Analysis

Circuit sensitivity can be defined as the influence of a change in the characteristics how much a particular circuit characteristic changes as a particular circuit-component value varies. This gives us an insight how creation parameters influence

in the response of a specific circuit. Here, AC sensitivities will be treated. For instances, one of the more popular notions used in circuit design is the andjoint network analysis [20], 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. Subsequently, 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 graph-based tool to compute symbolic sensitivities of analog ICs. In fact, graph-based approaches can also be applied to compute symbolic sensitivities of large analog ICs in a hierarchical fashion as shown in [25] for the DDD-approach.

Given the transfer function $H(s)$ seen as:

$$H(s) = \frac{N(s)}{D(s)} \tag{5.1}$$

where both $N(s)$ and $D(s)$ can be represented by a graph. The ac-sensitivity is given by the following normalized equation:

$$Sens(H(s), W) = \frac{W}{H(s)} \frac{\partial H(s)}{\partial W} \tag{5.2}$$

Substituting (5.1) in (5.2) and applying the chain rule we have:

$$Sens\left(H(s), W\right) = \left(\frac{W D(s)}{N(s)}\right) \left(\frac{\frac{\partial N(s)}{\partial W} D(s) - N(s) \frac{\partial D(s)}{\partial W}}{D^2(s)}\right) \tag{5.3}$$

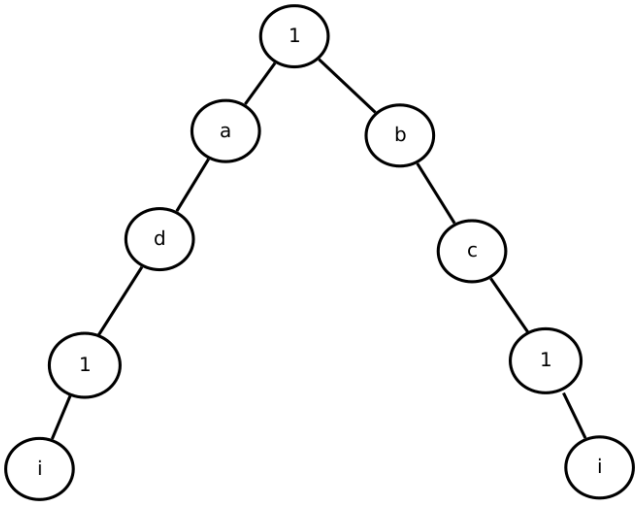


Figure 5.1: Resulting graph by applying the derivative

Regrouping similar terms in (5.3) we get to the expression shown in (5.4).

$$Sens(H(s), W) = W\left(\frac{1}{N(s)}\frac{\partial N(s)}{\partial W} - \frac{1}{D(s)}\frac{\partial D(s)}{\partial W}\right) \tag{5.4}$$

Equation (5.4) 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 tree can be replaced by 1 in the paths it is contained, while eliminating those paths that does not contain the symbolic variable [30].

If we want to get the derivative expression with respect to h in equation (2.2) for example, the resultant expanded graph would be as shown in Fig. 5.1

Symbol W	Symbolic Sens(H(s), W)	Numerical Sens(H(s), W)
gm_1	$\frac{\mathrm{gm_1\,r1}}{\mathrm{gm_1\,r1 - cgd_1\,r1}\,s}$	s=0, 1.00001
R_D	$\mathrm{r_1((gm_1 - cgd_1s)/(gds_1r_1 + s(cgd_1r_1 + c_1r_1) + 1) -}$ $\mathrm{((gm_1r_1 - cgd_1r_1s)(gds_1 + s(cgd_1 + c_1))/(gds_1r_1 +}$ $\mathrm{s(cgd_1r_1 + c_1r_1) + 1)^2)(gds_1r_1 + s(cgd_1r_1 + c_1r_1) + 1))/}$ $\mathrm{(gm_1r_1 - cgd_1r_1s)}$	s=0, 0.804

Table 5.1: Sensitivity with respect to gm_1 and R_D

Hence, the symbolic derivative expression for equation (2.2) becomes:

$$h = g + a \tag{5.5}$$

Symbolic sensitivity expression for a Common Source is shown. Sensitivity is formulated for gm_1 and R_D and the numerical evaluation is performed as shown in Table 5.1.

Chapter 6

Conclusions

In this work a new approach to the representation of symbolic determinants is proposed. This new technique extends over the previously known SOP graph representation with the aim to avoid redundant information.

Experiments were performed to test for correctness when performing the different analysis where good agreement between the numerical evaluation of the symbolic expressions and the results provided by an industry trusted numerical simulator ($HSpice^{TM}$) are presented.

Besides, the developed tool can run with the same netlist for input that $HSpice^{TM}$ handles avoiding the need for intermediary processing steps. With this approach it is possible to fully define the circuit variables in terms of node voltages or current branches. As well, noise figures can be computed in any node or branch.

Symbolic evaluation of output noise figures have already been reported in [44] and in [27]. The application of a new graph based technique as an efficient way to represent and solve for the determinant is joined and noise analysis is presented a a case study.

An additional advantage is the highly parallel nature of the resulting structure which comes handy when evaluating big matrices with todays multi-core computing systems. The formulated fully symbolic expressions represent the system to the highest accuracy bringing the possibility to perform with more detail analysis which benefit greatly by using symbolic expressions.

Chapter 7

Future Work

Several areas for implementation of the present tool are viable. Some of them which are of interest to the author are presented.

For instance, it is possible to implement new analysis to the resulting symbolic determinants like Model Order Reduction techniques previous to formulation, during formulation and post formulation.

The use of a symbolic processor as the sole tool for circuit analysis is impractical due to the size of current circuits. It is easier for the designer to *throw more CPU cores* into a problem and use the current numerical simulators. However, it's becoming of great interest the development of problem solving environments in which symbolic tools are used to abstract information about who are the trouble makers in a system along with the traditional approach of using numeric simulators. In this tenor, the present work is provided as a step towards the integration of a symbolic-numeric co-simulator.

A new area worth of exploring is the use of this simbolic tool in order to select the proper approach to the numeric solution, that is, tweak the numeric solution

algorithm implementation according to the circuit characteristic of interest in order to improve accuracy, reduce computer time or both.

Chapter 8

List of Publications

S. Rodriguez-Chavez, A.A. Palma-Rodriguez, E. Tlelo-Cuautle, S. X.-D. Tan, *Graph-based symbolic and symbolic sensitivity analysis of analog integrated circuits*, in Analog/RF and Mixed-Signal Circuit Systematic Design, M. Fakhfakh, E. Tlelo-Cuautle, R. Castro-Lpez (Eds.), Springer, 2012. In press

Santiago Rodriguez-Chavez, Esteban Tlelo-Cuautle, Adolfo A. Palma-Rodriguez, and Sheldon X.-D. Tan, *Symbolic DDD-based tool for the computation of noise in CMOS analog circuits*, International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), Playa del Carmen, Mxico, March 14-17, 2012.

Adolfo A. Palma-Rodriguez, Esteban Tlelo-Cuautle, Santiago Rodriguez-Chavez, and Sheldon X.-D. Tan, *DDD-Based Symbolic Sensitivity Analysis of Active Filters*, International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), Playa del Carmen, Mxico, March 14-17, 2012.

List of Figures

List of Tables

Appendix A

Symbolic Expressions

Symbolic expressions for amplifiers in Fig. 4.2

A.1 Common Source

For the differential pair shown in Fig. 4.2a.

A.1.1 Denominator D(s)

$(-1/r1 - s * cgd_1 - gds_1 - s * cdtot_1) * 1$

A.1.2 Numerator N(s)

$((-s * cgd_1 + gm_1) * 1)$

A.1.3 Transfer Function H(s) = N(s) / D(s)

$-(gm_1 - cgd_1 * s)/(gds_1 + cdtot_1 * s + cgd_1 * s + 1/r1)$

A.1.4 Sensitivity Sens(H(s), gm_1)

$gm_1/(gm_1 - cgd_1 * s)$

A.2 Differential Pair

For the differential pair shown in Fig. 4.2b.

A.2.1 Denominator D(s)

$$((-s*cgd_4+gm_4)*((-s*cgs_1-gm_1-gds_1-s*cdtot_1-gmb_1-s*cgs_2-gm_2-gds_2-s*cdtot_2-gmb_2-s*cgd_5-gds_5-s*cdtot_5)*(-s*cgd_4)*1*gev2_1*gev2_2+(-gds_2-s*cdtot_2)*(-gds_1-s*cdtot_1-gm_1-gmb_1)*1*gev2_1*gev2_2)+(gds_2+s*cdtot_2+gm_2+gmb_2)*((gds_1+s*cdtot_1)*(-s*cgd_4)*1*gev2_1*gev2_2+(-gds_2-s*cdtot_2)*(s*cgd_1+gds_1+s*cdtot_1+s*cgs_3+gds_3+s*cdtot_3+s*cgs_4+s*cgd_4+gm_3)*1*gev2_1*gev2_2)+(s*cgd_2+gds_2+s*cdtot_2+s*cgd_4+gds_4+s*cdtot_4)*((gds_1+s*cdtot_1)*(-gds_1-s*cdtot_1-gm_1-gmb_1)*1*gev2_1*gev2_2+(s*cgs_1+gm_1+gds_1+s*cdtot_1+gmb_1+s*cgs_2+gm_2+gds_2+s*cdtot_2+gmb_2+s*cgd_5+gds_5+s*cdtot_5)*(s*cgd_1+gds_1+s*cdtot_1+s*cgs_3+gds_3+s*cdtot_3+s*cgs_4+s*cgd_4+gm_3)*1*gev2_1*gev2_2))$$

A.2.2 Numerator N(s)

$$(((-s*cgd_4+gm_4)*((-s*cgs_1-gm_1)*(gds_1+s*cdtot_1+gm_1+gmb_1)*1*(-gev2_1)*(-gev1_2)+(-s*cgs_1-gm_1-gds_1-s*cdtot_1-gmb_1-s*cgs_2-gm_2-gds_2-s*cdtot_2-gmb_2-s*cgd_5-gds_5-s*cdtot_5)*(s*cgd_1-gm_1)*1*(-gev2_1)*(-gev1_2)+(-s*cgs_2-gm_2)*(-gds_1-s*cdtot_1-gm_1-gmb_1)*1*(-gev1_1)*gev2_2)+(gds_2+s*cdtot_2+gm_2+gmb_2)*((-s*cgs_1-gm_1)*(-s*cgd_1-gds_1-s*$$

$cdtot_1 - s*cgs_3 - gds_3 - s*cdtot_3 - s*cgs_4 - s*cgd_4 - gm_3) * 1 * (-gev2_1) * (-gev1_2) +$
$(gds_1 + s*cdtot_1) * (s*cgd_1 - gm_1) * 1 * (-gev2_1) * (-gev1_2) + (-s*cgs_2 - gm_2) *$
$(s*cgd_1 + gds_1 + s*cdtot_1 + s*cgs_3 + gds_3 + s*cdtot_3 + s*cgs_4 + s*cgd_4 + gm_3) *$
$1 * (-gev1_1) * gev2_2) + (-s*cgd_2 + gm_2) * ((gds_1 + s*cdtot_1) * (-gds_1 - s*cdtot_1 -$
$gm_1 - gmb_1) * 1 * (-gev1_1) * gev2_2 + (s*cgs_1 + gm_1 + gds_1 + s*cdtot_1 + gmb_1 + s*$
$cgs_2 + gm_2 + gds_2 + s*cdtot_2 + gmb_2 + s*cgd_5 + gds_5 + s*cdtot_5) * (s*cgd_1 + gds_1 + s*$
$cdtot_1 + s*cgs_3 + gds_3 + s*cdtot_3 + s*cgs_4 + s*cgd_4 + gm_3) * 1 * (-gev1_1) * gev2_2)))$

A.2.3 Transfer Function H(s) = N(s) / D(s)

$-((gev1_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*$
$s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+$
$cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev1_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+$
$gmb_1+cdtot_1*s))*(gm_2-cgd_2*s)+(gm_4-cgd_4*s)*(gev1_2*gev2_1*(gm_1+cgs_1*s)*$
$(gds_1+gm_1+gmb_1+cdtot_1*s)-gev1_2*gev2_1*(gm_1-cgd_1*s)*(gds_1+gds_2+gds_5+$
$gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*$
$s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))-(gev1_2*gev2_1*$
$(gm_1+cgs_1*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+$
$cgs_4*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+$
$cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gds_1+cdtot_1*s)*(gm_1-cgd_1*$
$s))*(gds_2+gm_2+gmb_2+cdtot_2*s))/((gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gm_1+$
$gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+$
$gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s))*(gm_4-cgd_4*s)+$
$(gev2_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+$
$cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*$

$s+cgd_5*s+cgs_1*s+cgs_2*s)-gev2_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s)-(gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+cdtot_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s))$

A.2.4 Sensitivity Sens(H(s), gm_1)

$(gm_1*(((gm_4-cgd_4*s)*(gev1_2*gev2_1*(gm_1+cgs_1*s)-gev1_2*gev2_1*(gm_1-cgd_1*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)-gev1_2*gev2_1*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+gev1_2*gev2_1*(gds_1+gm_1+gmb_1+cdtot_1*s))-(gev1_2*gev2_1*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gds_1+cdtot_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s)+(gev1_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gds_1+cdtot_1*s))*(gm_2-cgd_2*s))/((gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s))*(gm_4-cgd_4*s)+(gev2_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev2_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s)-(gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+cdtot_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s))-(((gev2_1*gev2_2*(gds_2+cdtot_2*s)+cgd_4*gev2_1*gev2_2*s)*(gm_4-cgd_4*s)+(gev2_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev2_1*gev2_2*(gds_1+cdtot_1*s))*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+$

$$cgd_2*s+cgd_4*s))*((gev1_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev1_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_2-cgd_2*s)+(gm_4-cgd_4*s)*(gev1_2*gev2_1*(gm_1+cgs_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)-gev1_2*gev2_1*(gm_1-cgd_1*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))-(gev1_2*gev2_1*(gm_1+cgs_1*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gds_1+cdtot_1*s)*(gm_1-cgd_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s)))/((gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s))*(gm_4-cgd_4*s)+(gev2_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev2_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s)-(gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+cdtot_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s))^2)*((gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s))*(gm_4-cgd_4*s)+(gev2_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev2_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*$$

$(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s)-(gev2_1*gev2_2*(gds_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gds_1+cdtot_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s)))/((gev1_1*gev2_2*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)-gev1_1*gev2_2*(gds_1+cdtot_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_2-cgd_2*s)+(gm_4-cgd_4*s)*(gev1_2*gev2_1*(gm_1+cgs_1*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)-gev1_2*gev2_1*(gm_1-cgd_1*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))-(gev1_2*gev2_1*(gm_1+cgs_1*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+gev1_1*gev2_2*(gm_2+cgs_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gds_1+cdtot_1*s)*(gm_1-cgd_1*s))*(gds_2+gm_2+gmb_2+cdtot_2*s))$

A.3 Three Stages Uncompensated OTA

For the differential pair shown in Fig. 4.2c

A.3.1 Denominator D(s)

$((s*cgs_8+gm_8)*((-gds_2-s*cdtot_2)*((-s*cgd_1-gds_1-s*cdtot_1-s*cgs_3-gds_3-s*cdtot_3-s*cgs_4-s*cgd_4-gm_3)*(-gds_2-s*cdtot_2-gm_2-gmb_2)*(-s*cgs_8)*1*gev2_1*gev2_2+(-gds_1-s*cdtot_1-gm_1-gmb_1)*(-s*cgd_4+gm_4)*(-s*cgs_8)*1*gev2_1*gev2_2)+(gds_1+s*cdtot_1)*(s*cgd_4*(-gds_2-s*cdtot_2-gm_2-gmb_2)*(-s*cgs_8)*1*gev2_1*gev2_2+(-gds_1-s*cdtot_1-gm_1-gmb_1)*(s*cgd_2+gds_2+s*cdtot_2+s*cgd_4+gds_4+s*cdtot_4+s*cgs_6+s*cgd_6)*(-s*cgs_8)*1*gev2_1*gev2_2)+(s*cgs_1+$

$$gm_1+gds_1+s*cdtot_1+gmb_1+s*cgs_2+gm_2+gds_2+s*cdtot_2+gmb_2+s*cgd_5+gds_5+s*cdtot_5)*(s*cgd_4*(-s*cgd_4+gm_4)*(-s*cgs_8)*1*gev2_1*gev2_2+(s*cgd_1+gds_1+s*cdtot_1+s*cgs_3+gds_3+s*cdtot_3+s*cgs_4+s*cgd_4+gm_3)*(s*cgd_2+gds_2+s*cdtot_2+s*cgd_4+gds_4+s*cdtot_4+s*cgs_6+s*cgd_6)*(-s*cgs_8)*1*gev2_1*gev2_2))+(s*cl+s*cgs_8+gm_8+gds_8+s*cdtot_8+gmb_8+s*cgd_9+gds_9+s*cdtot_9)*((gds_2+s*cdtot_2)*((s*cgd_1+gds_1+s*cdtot_1+s*cgs_3+gds_3+s*cdtot_3+s*cgs_4+s*cgd_4+gm_3)*(gds_2+s*cdtot_2+gm_2+gmb_2)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2+(gds_1+s*cdtot_1+gm_1+gmb_1)*(s*cgd_4-gm_4)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2)+(-gds_1-s*cdtot_1)*((-s*cgd_4)*(gds_2+s*cdtot_2+gm_2+gmb_2)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2+(gds_1+s*cdtot_1+gm_1+gmb_1)*((-s*cgd_6)*(s*cgd_6-gm_6)*1*gev2_1*gev2_2+(-s*cgd_2-gds_2-s*cdtot_2-s*cgd_4-gds_4-s*cdtot_4-s*cgs_6-s*cgd_6)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2))+(-s*cgs_1-gm_1-gds_1-s*cdtot_1-gmb_1-s*cgs_2-gm_2-gds_2-s*cdtot_2-gmb_2-s*cgd_5-gds_5-s*cdtot_5)*((-s*cgd_4)*(s*cgd_4-gm_4)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2+(-s*cgd_1-gds_1-s*cdtot_1-s*cgs_3-gds_3-s*cdtot_3-s*cgs_4-s*cgd_4-gm_3)*((-s*cgd_6)*(s*cgd_6-gm_6)*1*gev2_1*gev2_2+(-s*cgd_2-gds_2-s*cdtot_2-s*cgd_4-gds_4-s*cdtot_4-s*cgs_6-s*cgd_6)*(-s*cgd_6-gds_6-s*cdtot_6-s*cgd_7-gds_7-s*cdtot_7-s*cgs_8-s*cgd_8)*1*gev2_1*gev2_2))))$$

A.3.2 Numerator N(s)

$$((s*cgs_8+gm_8)*((gds_1+s*cdtot_1)*((-gds_1-s*cdtot_1-gm_1-gmb_1)*(s*cgd_2-gm_2)*(s*cgd_6-gm_6)*1*(-gev2_1)*(-gev1_2)+(-s*cgd_1+gm_1)*(gds_2+s*cdtot_2+$$

$$gm_2 + gmb_2) * (s * cgd_6 - gm_6) * 1 * (-gev1_1) * gev2_2) + (s * cgs_1 + gm_1 + gds_1 + s * cdtot_1 + gmb_1 + s * cgs_2 + gm_2 + gds_2 + s * cdtot_2 + gmb_2 + s * cgd_5 + gds_5 + s * cdtot_5) * ((s * cgd_1 + gds_1 + s * cdtot_1 + s * cgs_3 + gds_3 + s * cdtot_3 + s * cgs_4 + s * cgd_4 + gm_3) * (s * cgd_2 - gm_2) * (s * cgd_6 - gm_6) * 1 * (-gev2_1) * (-gev1_2) + (-s * cgd_1 + gm_1) * (s * cgd_4 - gm_4) * (s * cgd_6 - gm_6) * 1 * (-gev1_1) * gev2_2) + (s * cgs_2 + gm_2) * ((s * cgd_1 + gds_1 + s * cdtot_1 + s * cgs_3 + gds_3 + s * cdtot_3 + s * cgs_4 + s * cgd_4 + gm_3) * (gds_2 + s * cdtot_2 + gm_2 + gmb_2) * (s * cgd_6 - gm_6) * 1 * (-gev2_1) * (-gev1_2) + (gds_1 + s * cdtot_1 + gm_1 + gmb_1) * (s * cgd_4 - gm_4) * (s * cgd_6 - gm_6) * 1 * (-gev2_1) * (-gev1_2)) + (-s * cgs_1 - gm_1) * ((s * cgd_1 + gds_1 + s * cdtot_1 + s * cgs_3 + gds_3 + s * cdtot_3 + s * cgs_4 + s * cgd_4 + gm_3) * (gds_2 + s * cdtot_2 + gm_2 + gmb_2) * (s * cgd_6 - gm_6) * 1 * (-gev1_1) * gev2_2 + (gds_1 + s * cdtot_1 + gm_1 + gmb_1) * (s * cgd_4 - gm_4) * (s * cgd_6 - gm_6) * 1 * (-gev1_1) * gev2_2)))$$

A.3.3 Transfer Function H(s) = N(s) / D(s)

$$((gm_8 + cgs_8 * s) * ((gev1_1 * gev2_2 * (gm_6 - cgd_6 * s) * (gds_2 + gm_2 + gmb_2 + cdtot_2 * s) * (gds_1 + gds_3 + gm_3 + cdtot_1 * s + cdtot_3 * s + cgd_1 * s + cgd_4 * s + cgs_3 * s + cgs_4 * s) - gev1_1 * gev2_2 * (gm_4 - cgd_4 * s) * (gm_6 - cgd_6 * s) * (gds_1 + gm_1 + gmb_1 + cdtot_1 * s)) * (gm_1 + cgs_1 * s) - (gev1_2 * gev2_1 * (gm_2 - cgd_2 * s) * (gm_6 - cgd_6 * s) * (gds_1 + gds_3 + gm_3 + cdtot_1 * s + cdtot_3 * s + cgd_1 * s + cgd_4 * s + cgs_3 * s + cgs_4 * s) - gev1_1 * gev2_2 * (gm_1 - cgd_1 * s) * (gm_4 - cgd_4 * s) * (gm_6 - cgd_6 * s)) * (gds_1 + gds_2 + gds_5 + gm_1 + gm_2 + gmb_1 + gmb_2 + cdtot_1 * s + cdtot_2 * s + cdtot_5 * s + cgd_5 * s + cgs_1 * s + cgs_2 * s) + (gev1_2 * gev2_1 * (gm_6 - cgd_6 * s) * (gds_2 + gm_2 + gmb_2 + cdtot_2 * s) * (gds_1 + gds_3 + gm_3 + cdtot_1 * s + cdtot_3 * s + cgd_1 * s + cgd_4 * s + cgs_3 * s + cgs_4 * s) - gev1_2 * gev2_1 * (gm_4 - cgd_4 * s) * (gm_6 - cgd_6 * s) * (gds_1 + gm_1 + gmb_1 + cdtot_1 * s)) * (gm_2 + cgs_2 * s) + (gev1_2 * gev2_1 * (gm_2 - cgd_2 * s) * (gm_6 - cgd_6 * s) * (gds_1 + gm_1 + gmb_1 + cdtot_1 * s) - gev1_1 * gev2_2 * (gm_1 - cgd_1 * s) * (gm_6 - cgd_6 * s) * (gds_2 + gm_2 +$$

$$gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))/(((gev2_1*gev2_2*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev2_1*gev2_2*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_2+cdtot_2*s)-((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+cdtot_1*s))*(gds_8+gds_9+gm_8+gmb_8+cdtot_8*s+cdtot_9*s+cgd_9*s+cgs_8*s+cl*s)-(gm_8+cgs_8*s)*((cgs_8*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-cgs_8*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+cdtot_2*s)-(cgs_8*gev2_1*gev2_2*s*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gm_4-cgd_4*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(cgs_8*gev2_1*gev2_2*s*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*$$

$s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))$

A.3.4 Sensitivity Sens(H(s), gm_1)

$(gm_1*(((gev2_1*gev2_2*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev2_1*gev2_2*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_2+cdtot_2*s)-((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+cdtot_1*s))*(gds_8+gds_9+gm_8+gmb_8+cdtot_8*s+cdtot_9*s+cgd_9*s+cgs_8*s+cl*s)-(gm_8+cgs_8*s)*((cgs_8*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-cgs_8*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+cdtot_2*s)-(cgs_8*gev2_1*gev2_2*s*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gm_4-cgd_4*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+$

$$gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(cgs_8*gev2_1*gev2_2*s*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))*(((gm_8+cgs_8*s)*((gev1_2*gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)-gev1_1*gev2_2*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)-gev1_2*gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+gev1_1*gev2_2*(gm_1-cgd_1*s)*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)-gev1_1*gev2_2*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gm_1+cgs_1*s)-gev1_2*gev2_1*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gm_2+cgs_2*s)+gev1_1*gev2_2*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+gev1_1*gev2_2*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)))/(((gev2_1*gev2_2*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev2_1*gev2_2*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_2+cdtot_2*s)-((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+$$

$$cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+cdtot_1*s))*(gds_8+gds_9+gm_8+gmb_8+cdtot_8*s+cdtot_9*s+cgd_9*s+cgs_8*s+cl*s)-(gm_8+cgs_8*s)*((cgs_8*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-cgs_8*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+cdtot_2*s)-(cgs_8*gev2_1*gev2_2*s*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gm_4-cgd_4*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(cgs_8*gev2_1*gev2_2*s*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))-(((gm_8+cgs_8*s)*(cgs_8*gev2_1*gev2_2*s*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-cgs_8*gev2_1*gev2_2*s*(gds_1+cdtot_1*s)*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gm_4-cgd_4*s)+cgs_8*gev2_1*gev2_2*s*(gds_2+cdtot_2*s)*(gm_4-cgd_4*s))-((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-(gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+cdtot_1*s)+gev2_1*gev2_2*(gds_2+cdtot_2*s)*$$

$$(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_4*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_8+gds_9+gm_8+gmb_8+cdtot_8*s+cdtot_9*s+cgd_9*s+cgs_8*s+cl*s))*(gm_8+cgs_8*s)*((gev1_1*gev2_2*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_1+cgs_1*s)-(gev1_2*gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gm_1-cgd_1*s)*(gm_4-cgd_4*s)*(gm_6-cgd_6*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(gev1_2*gev2_1*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_2+cgs_2*s)+(gev1_2*gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)-gev1_1*gev2_2*(gm_1-cgd_1*s)*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))/(((gev2_1*gev2_2*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev2_1*gev2_2*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_2+cdtot_2*s)-((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+$$

$$cgd_5*s+cgs_1*s+cgs_2*s)+((gev2_1*gev2_2*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s)+cgd_6*gev2_1*gev2_2*s*(gm_6-cgd_6*s))*(gds_1+gm_1+gmb_1+cdtot_1*s)+cgd_4*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_6+gds_7+cdtot_6*s+cdtot_7*s+cgd_6*s+cgd_7*s+cgd_8*s+cgs_8*s))*(gds_1+cdtot_1*s))*(gds_8+gds_9+gm_8+gmb_8+cdtot_8*s+cdtot_9*s+cgd_9*s+cgs_8*s+cl*s)-(gm_8+cgs_8*s)*((cgs_8*gev2_1*gev2_2*s*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-cgs_8*gev2_1*gev2_2*s*(gm_4-cgd_4*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gds_2+cdtot_2*s)-(cgs_8*gev2_1*gev2_2*s*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gm_4-cgd_4*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(cgs_8*gev2_1*gev2_2*s*(gds_1+gm_1+gmb_1+cdtot_1*s)*(gds_2+gds_4+cdtot_2*s+cdtot_4*s+cgd_2*s+cgd_4*s+cgd_6*s+cgs_6*s)+cgd_4*cgs_8*gev2_1*gev2_2*s^2*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))^2))/((gm_8+cgs_8*s)*((gev1_1*gev2_2*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_1+cgs_1*s)-(gev1_2*gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_1*gev2_2*(gm_1-cgd_1*s)*(gm_4-cgd_4*s)*(gm_6-cgd_6*s))*(gds_1+gds_2+gds_5+gm_1+gm_2+gmb_1+gmb_2+cdtot_1*s+cdtot_2*s+cdtot_5*s+cgd_5*s+cgs_1*s+cgs_2*s)+(gev1_2*gev2_1*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s)*(gds_1+gds_3+gm_3+cdtot_1*s+cdtot_3*s+cgd_1*s+cgd_4*s+cgs_3*s+cgs_4*s)-gev1_2*gev2_1*(gm_4-cgd_4*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s))*(gm_2+cgs_2*s)+(gev1_2*$$

$$gev2_1*(gm_2-cgd_2*s)*(gm_6-cgd_6*s)*(gds_1+gm_1+gmb_1+cdtot_1*s)-gev1_1*gev2_2*$$
$$(gm_1-cgd_1*s)*(gm_6-cgd_6*s)*(gds_2+gm_2+gmb_2+cdtot_2*s))*(gds_1+cdtot_1*s)))$$

Bibliography

[1] HSPICE: Quick Reference Guide ; HSPICE Version. v. 92. Meta-Software (1992)

[2] Agnew, D.: Efficient use of the Hessian matrix for circuit optimization. Circuits and Systems, IEEE Transactions on **25**(8), 600–608 (1978). DOI 10.1109/TCS.1978.1084520

[3] Bryant, R.E.: Boolean analysis of MOS circuits. Computer Science Department p. 187 (1987)

[4] Carlin, H.: Singular Network Elements. Circuit Theory, IEEE Transactions on **11**(1), 67–72 (1964). DOI 10.1109/TCT.1964.1082264

[5] Dordevic, S., Petkovi, P.M.: Symbolic-numeric co-simulation of large analogue circuits. In: Microelectronics, 2002. MIEL 2002. 23rd International Conference on, vol. 2, pp. 639–642. IEEE (2002)

[6] E. Sanches-Sinencio: Low voltage Amplifier Design Techniques. Tutorial in IEEE MWSCAS (2009)

[7] Fakhfakh, M., Tlelo-Cuautle, E., Fernández, F.V.: Design of Analog Circuits through Symbolic Analysis. Bentham Science Publishers Ltd. (2012)

[8] Fernández, F.V., Rodríguez-Vázquez, A., Huertas, J.L.: A tool for symbolic analysis of analog integrated circuits including pole/zero extraction. In: Proc. ECCTD91 (1991)

[9] Fernández, F.V., Rodríguez-Vázquez, A., Huertas, J.L.: Interactive AC modeling and characterization of analog circuits via symbolic analysis. Analog Integrated Circuits and Signal Processing **1**(3), 183–208 (1991)

[10] Fernández F. V. Rodríguez- Vázquez, A., Huertas, J.L.: A tool for symbolic analysis of analog integrated circuits including pole/zero extraction. In: Proc. 10th European Conf. on Circuit Theory and Design 2, pp. 752–761 (1991)

[11] Gallopoulos, S., Houstis, E., Rice, J.: Future research directions in problem solving environments for computational science (1992)

[12] Geiger, R.L., Allen, P.E., Strader, N.R.: VLSI design techniques for analog and digital circuits. McGraw-Hill series in electrical engineering. McGraw-Hill Book Co. (1990)

[13] Gielen, G., Wambacq, P., Sansen, W.M.: Symbolic analysis methods and applications for analog circuits: a tutorial overview. Proceedings of the IEEE **82**(2), 287–304 (1994). DOI 10.1109/5.265355

[14] Gielen, G.G.E., Walscharts, H.C.C., Sansen, W.M.C.: ISAAC: a symbolic simulator for analog integrated circuits. Solid-State Circuits, IEEE Journal of **24**(6), 1587–1597 (1989). DOI 10.1109/4.44994

[15] Gómez, I.G.: Optimización multiobjetivo de circuitos electrónicos aplicando al-

goritmos evolutivos. Phd dissertation, Instituto Nacional de Astrofísica, Óptica y Electrónica (2012)

[16] Gray, P.R., Meyer, R.G.: Analysis and design of analog integrated circuits. John Wiley & Sons, Inc. (1990)

[17] Hassoun, M.M., Lin, P.M.: A new network approach to symbolic simulation of large-scale networks. In: Circuits and Systems, 1989., IEEE International Symposium on, pp. 806 –809 vol.2 (1989). DOI 10.1109/ISCAS.1989.100473

[18] Huang, Q., Piazza, F., Orsatti, P., Ohguro, T.: The impact of scaling down to deep submicron on CMOS RF circuits. Solid-State Circuits, IEEE Journal of **33**(7), 1023–1036 (1998)

[19] Huelsman, L.P.: Personal computer symbolic analysis program for undergraduate engineering courses. In: Circuits and Systems, 1989., IEEE International Symposium on, pp. 798–801. IEEE (1989)

[20] J. Vlach, K.S.: Computer Method for Circuit Analysis and Design, 2 edn. International Thomson Publishing (1994)

[21] Katzenelson, J., Unikovski, A.: Symbolic-numeric circuit analysis or symbolic circuit analysis with online approximations. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on **46**(1), 197–207 (1999)

[22] Konczykowska, A., Bon, M.: Automated design software for switched-capacitor IC's with symbolic simulator SCYMBAL. In: Proceedings of the 25th ACM/IEEE Design Automation Conference, DAC '88, pp. 363–368. IEEE Computer Society Press, Los Alamitos, CA, USA (1988)

[23] Kuo, J.B., Lou, J.H., Luo, J.H.: Low-voltage CMOS VLSI circuits. Wiley-Interscience publication. John Wiley (1999)

[24] Laker, K.R., Sansen, W.M.C.: Design of analog integrated circuits and systems. McGraw-Hill series in electrical and computer engineering: Electronics and VLSI circuits. McGraw-Hill (1994)

[25] Li, X., Xu, H., Shi, G., Tai, A.: Hierarchical symbolic sensitivity computation with applications to large amplifier circuit design. In: Circuits and Systems (ISCAS), 2011 IEEE International Symposium on, pp. 2733–2736 (2011). DOI 10.1109/ISCAS.2011.5938170

[26] Manetti, S.: New approach to automatic symbolic analysis of electric circuits. Circuits, Devices and Systems, IEE Proceedings G **138**(1), 22–28 (1991)

[27] Martínez-romero, E., Tlelo-cuautle, E., Sánchez-lópez, C., Tan, S.X.: Symbolic Noise Analysis of Low Voltage Amplifiers by Using Nullors **1**, 4–8 (2010)

[28] Martínez-Romero, E., Tlelo-Cuautle, E., Sánchez-López, C., Tan, S.X.D.: Symbolic Noise Analysis of Low Voltage Amplifiers by Using Nullors. In: Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD), 2010 XIth International Workshop on, pp. 1–5. IEEE (2010)

[29] Moore, G.: Moore s Law : Raising the Bar (February) (2003)

[30] Palma-Rodriguez, A.A., Tlelo-Cuautle, E., Rodriguez-Chavez, S., Tan, S.X.: DDD-based symbolic sensitivity analysis of active filters. In: Devices, Circuits and Systems (ICCDCS), 2012 8th International Caribbean Conference on, pp. 1–4 (2012). DOI 10.1109/ICCDCS.2012.6188913

[31] Patra, P., Kumaravel, S., Venkatramani, B.: An enhanced fully differential Recyclic Folded Cascode OTA. In: Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on, pp. 1–5 (2012). DOI 10.1109/SSD.2012.6197977

[32] Pi, T., Shi, C.J.R.: Analog-testability analysis by determinant-decision-diagrams based symbolic analysis. In: Proceedings of the 2000 Asia and South Pacific Design Automation Conference, pp. 541–546. ACM (2000)

[33] Razavi, B.: RF Microelectronics. Prentice Hall International Series in the Physical and Chemical Engineering Sciences. Pearson Education (2011)

[34] Rodriguez-Chavez, S., Tlelo-Cuautle, E., Palma-Rodriguez, A.A., Tan, S.D.: Symbolic DDD-based tool for the computation of noise in CMOS analog circuits. In: Devices, Circuits and Systems (ICCDCS), 2012 8th International Caribbean Conference on, pp. 1–4 (2012). DOI 10.1109/ICCDCS.2012.6188912

[35] Sánchez López, C.: Optimizacin multiobjetivo de circuitos electrnicos aplicando algoritmos evolutivos. Phd dissertation, Instituto Nacional de Astrofísica, Óptica y Electrónica (2012)

[36] Sanchez-Lopez, C., Fernandez, F.V., Tlelo-Cuautle, E., Tan, S.X.: Pathological Element-Based Active Device Models and Their Application to Symbolic Analysis. Circuits and Systems I: Regular Papers, IEEE Transactions on **58**(6), 1382–1395 (2011). DOI 10.1109/TCSI.2010.2097696

[37] Sánchez-Sinencio, E., Andreou, A.G., Society, I.S.S.C.: Low-voltage/low-power

integrated circuits and systems: low-voltage mixed-signal ciruits. IEEE Press series on microelectronic systems. IEEE Press (1999)

[38] Sansen, W., Glelen, G., Walscharts, H.: A symbolic simulator for analog circuits. In: Solid-State Circuits Conference, 1989. Digest of Technical Papers. 36th ISSCC., 1989 IEEE International, pp. 204–205 (1989). DOI 10.1109/ISSCC.1989.48261

[39] Seda, S.J., Degrauwe, M.G.R., Fichtner, W.: A symbolic analysis tool for analog circuit design automation. In: Computer-Aided Design, 1988. ICCAD-88. Digest of Technical Papers., IEEE International Conference on, pp. 488–491 (1988). DOI 10.1109/ICCAD.1988.122555

[40] Seda, S.J., Degrauwe, M.G.R., Fichtner, W.: Lazy-expansion symbolic expression approximation in SYNAP. In: Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on, pp. 310–317 (1992). DOI 10.1109/ICCAD.1992.279355

[41] Shi, C.J.R., Tan, X.: Symbolic analysis of large analog circuits with determinant decision diagrams. In: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, pp. 366–373. IEEE Computer Society (1997)

[42] Tan, S.X.D.: Symbolic Analysis by Determinant Decision Diagrams and Applications. In: Design of Analog Circuits through SYMBOLIC ANALYSIS. Bentham Science Publishers Ltd. (2012)

[43] Tan, X.D.: Symbolic analysis of large analog circuits with determinant decision diagrams. Ph.D. thesis, University of Iowa (1999)

[44] Tlelo-Cuautle, E., Sanchez-Lopez, C.: Symbolic computation of NF of transistor circuits. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences **87**(9), 2420–2425 (2004)

[45] Tsividis, Y., McAndrew, C.: Operation and Modeling of the Mos Transistor. The Oxford Series in Electrical and Computer Engineering. Oxford University Press (2010)

[46] Wang, H.Y., Huang, W.C., Chiang, N.H.: Symbolic Nodal Analysis of Circuits Using Pathological Elements. Circuits and Systems II: Express Briefs, IEEE Transactions on **57**(11), 874–877 (2010). DOI 10.1109/TCSII.2010.2082930

[47] Wierzba, G.M., Srivastava, A., Joshi, V., Noren, K.V., Svoboda, J.A.: Sspice-a symbolic SPICE program for linear active circuits. In: Circuits and Systems, 1989., Proceedings of the 32nd Midwest Symposium on, pp. 1197 –1201 vol.2 (1989). DOI 10.1109/MWSCAS.1989.102070

[48] Van der Ziel, A.: Noise sources, characterization, measurement. Prentice-Hall information and system sciences series. Prentice-Hall (1970)