

Parasitic-Aware Hierarchical Symbolic Performance Modeling for Layout-Inclusive Synthesis of Large Analog Circuits^{*}

ABSTRACT

The primary focus of this work is on the generation of layout-aware symbolic performance models (SPMs), for parasitic-inclusive large analog circuits, by using exact hierarchical symbolic analysis. Two problems arise while generating these SPMs. The first problem is the symbolic analysis of large networks. We propose a new hierarchical symbolic analysis technique, that takes the modular approach of layout-generation and uses it for large circuits. The core of this algorithm is a novel idea where transfer functions are synthesized for a general interconnection template of two subcircuits. The second problem deals with the generation and partitioning of circuit topologies, that include all parasitics generated in a synthesis run. We propose efficient techniques for parasitic-inclusive topology generation and partitioning. In this paper we also use the SPMs, in layout-inclusive synthesis of a large analog circuit, to overcome important deficiencies in traditional analog synthesis. The accuracy and effectiveness of these SPMs has also been demonstrated.

1. INTRODUCTION

The process of determining numerical values for the unsized elements of an analog circuit, while satisfying a set of performance constraints, is called circuit synthesis. In traditional analog synthesis [3], a combinatorial optimization algorithm generates several alternative sets of component sizes. Each set of sizes is used for performance estimation by a numerical circuit simulator. The process converges when all constraints are satisfied. This synthesis approach suffers from two important shortcomings - a) unawareness to the layout parasitics and b) computationally expensive numerical analysis for performance estimation. Not estimating layout-effects during optimization can lead to the failure of the synthesized circuit in the post-layout verification stage. Numerical simulation inside the synthesis loop results in large per-iteration time, as well as, large synthesis time.

The aforementioned deficiencies in analog synthesis can be alleviated by using layout-inclusive techniques along with symbolic analysis based performance models. *Layout-inclusive* synthesis is a circuit sizing method where the set of parameter values proposed by the optimizer are used to either generate or instantiate a complete layout inside the synthesis loop. This technique is also called a *layout-in-loop* approach. *Symbolic performance models (SPMs)* are symbolic equations in terms of circuit parameters, which represent the characteristics of an analog circuit [7, 5]. SPMs are built using symbolic transfer functions obtained by symbolic circuit analysis.

In [5] a synthesis method combining the concepts of layout-inclusion and SPMs is presented. Layout generation is included in the synthesis loop by using a parameterized layout generator. SPMs are generated using the symbolic analysis method based on element-coefficient diagrams (ECDs). ECD's are graph-based data structures which store exact (no approximations) transfer functions expressions compactly and were first presented in [7]. For use in synthesis the ECDs are converted into C++ code and compiled for extremely fast evaluation. In such a synthesis method parasitic-inclusive topologies have to be

used for ECD generation, in order to capture the layout effects. Even for medium sized analog circuits, due to inclusion of parasitic elements, topologies becomes very complex. The synthesis methodology in works well for small and medium sized analog circuits, but as the size of circuits increase it is difficult to compile and eventually to generate the ECD code. For even larger circuits symbolic technique fails to generate the ECDs.

To overcome the limitation on the size of circuits, a hierarchical symbolic analysis approach has to be used. Techniques involving approximation/simplification cannot be used because in synthesis the accuracy of performance models is of paramount importance. Two relevant hierarchical methods are: DDD-based [6] and structural regularity based [2]. The first two approaches are not suitable for this application because the size of circuits become very large due to the inclusion of parasitics. A method, where subcircuits transfer functions (TFs) are combined to yield the TFs for a higher level circuit, is preferable. Only then can the modular properties of topologies obtained from layouts be exploited. In [2] a decomposition-based approach is proposed, but breaks down the circuit to the smallest, most basic, elements. This approach does not correspond directly to the layout, which uses the common analog modules. A technique based on decomposition and circuit function generation is presented [1], but is limited to small circuits because it does not present a formal method which can be used for large circuits.

In this paper we present a new hierarchical symbolic analysis method based on the use of a *general interconnection template (GIT)*. The GIT represents all possible ways to connect two subcircuits. A complete analysis of the GIT is done using the principles of transfer function synthesis. The transfer functions which completely characterize the GIT are thus obtained from the transfer functions of the two general subcircuits. The complete analysis of the GIT is not presented in the paper but the representative equations detailing the process have been shown. This hierarchical analysis method takes as input a partitioned circuit. The parasitic-inclusive circuit is first generated using techniques described in the following section. Then the circuit is partitioned according to a module-based approach. A manual approach or an algorithmic approach can also be used for partitioning.

The rest of the paper is organized as follows. In Section 2, we describe the techniques to generate parasitic-inclusive netlists, that contain superset of all parasitics generated in any synthesis run. The partitioning techniques for parasitic-inclusive topologies are described in Section 3. Section 4 presents the GIT and the hierarchical symbolic analysis technique using based on the transfer function synthesis. The implementation framework has also been presented. The symbolic performance modeling method is described briefly in Section 5. The synthesis framework is described Section 6. Experimental results are discussed in Section 7. A summary of conclusions and future work is described in Section 6.

2. PARASITIC-INCLUSIVE TOPOLOGY GENERATION

The first task is the generation of circuit topologies, which include all the parasitics that can possibly be generated during a synthesis run. As component sizes vary during synthesis, the layout geometry varies between iterations. This variation may generate varying sets of par-

^{*}This work is supported in part by DARPA/MTO NeoCAD program under grant number F333615-01-C-1977 monitored by the Air Force Research Laboratory.

asitic elements (resistances and capacitances) in each iteration. Let $C(R)$ be the set of all parasitic capacitances (resistances) ever appearing in an extracted circuit. Some elements of $C(R)$ might be missing (set to zero) in some instances of the extracted circuits. However, an SPM including all potential parasitic capacitances (C) and resistances (R), must be pre-generated. Techniques to do so are described next.

2.1 Analysis of the Layout Template

This technique relies on using the template used for layout generation in the synthesis loop to generate the parasitic-inclusive topology.

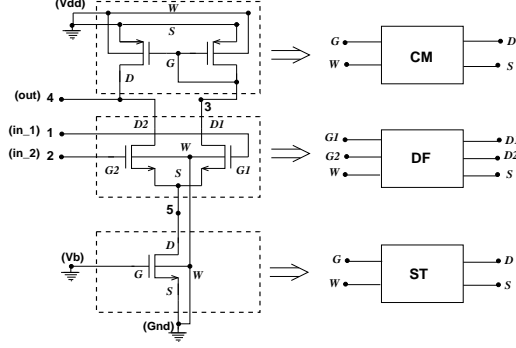


Figure 1: Diff. Pair with Current Mirror & Current Source

The first step is to select the modules essential for building the circuit. In the circuit shown in Figure 1, the three module required are: differential pair (DP), current mirror (CM) and a single transistor (ST). Each module is characterized by a list of nodes. The differential pair has the following input/output nodes: G_1, G_2, D_1, D_2, S, W .

The nodes of the selected modules are assigned physical node names from the circuit schematic. For symbolic analysis, each DC voltage source in the circuit is shorted and DC current sources are open circuited. The nodes S and W are powered by a DC voltage source V_{dd} , but in the circuit schematic (Figure 1) they are shown to be connected to ground directly. The final assignment of nodes for the three modules are thus: DP ($G_1 = 1, G_2 = 2, D_1 = 3, D_2 = 4, S = 5, W = 0$), CM ($G = 3, D = 4, S = 0, W = 0$) and ST ($G = 0, D = 5, S = 0, W = 0$)

The assignment of nodes from the circuit topology to modules, enables the generation of the set of actual possible parasitics. Let the original set of parasitics associated with any module X be referred to as $CA_{M,X}$. After the circuit-node assignment this set of parasitics is reduced and yields the set $CR_{M,X}$. Set $CA_{M,CM}$ is the list of parasitics generated by an extraction of the current mirror module.

$$CA_{M,CM} = \{C_{S,W}, C_{D,G}, C_{G,S}, C_{G,W}, C_{D,W}, C_{D,S},$$

$$C_{D,GND}, C_{G,GND}, C_{S,GND}, C_{W,GND}\}$$

The node GND is eventually replaced by the ground (0) node. After replacing the module nodes by circuit nodes:

$$CA_{M,CM} = \{C_{0,0}, C_{4,3}, C_{3,0}, C_{3,0}, C_{4,0}, C_{4,0}, C_{4,0}, C_{3,0}, C_{0,0}, C_{0,0}\}$$

By removing invalid and redundant capacitances a reduced set of capacitance is obtained:

$$CR_{M,CM} = \{C_{4,3}, C_{4,0}, C_{3,0}\}$$

The same transformations are applied to the original capacitance set of each module and similarly $CR_{M,DP}$ and $CR_{M,ST}$ are obtained. The reduced set of module parasitics for the whole circuit is obtained thus:

$$CR_M = CR_{M,CM} \cup CR_{M,DP} \cup CR_{M,ST} \\ = \{C_{4,3}, C_{4,0}, C_{3,0}, C_{1,5}, C_{5,3}, C_{2,4}, C_{4,5}, C_{1,0}, C_{2,0}, C_{5,0}, C_{1,3}, C_{2,5}\}$$

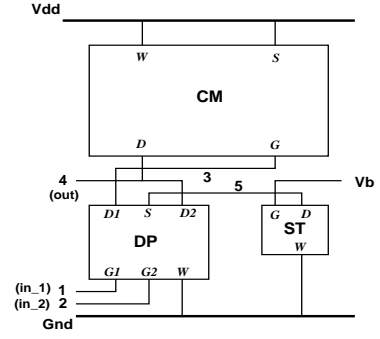


Figure 2: An Example Layout Template

Once the reduced set of module parasitics for the circuit have been determine, the focus shifts to interconnect parasitics. Figure 2 shows the template used for layout generation for the circuit in Figure 1. The layout template shows the presence of three channels for interconnects. First, we consider the case where only parasitic capacitances are extracted. The method described in [5] to obtain the parasitic capacitances of interconnects generates the following sets of parasitics. The set of interconnect parasitics for channels 0, 1 and 2, after assigning the nodes V_{dd}, V_b and Gnd to 0, are:

$$C_{IC_0} = \{C_{1,2}, C_{1,0}, C_{2,0}\}; C_{IC_1} = \{C_{3,4}, C_{3,5}, C_{4,5}\}; C_{IC_2} = \phi$$

Therefore the complete set of interconnect parasitics is:

$$C_{IC} = C_{IC_0} \cup C_{IC_1} \cup C_{IC_2} = \{C_{1,2}, C_{1,0}, C_{2,0}, C_{3,4}, C_{3,5}, C_{4,5}\}$$

Most of the interconnect parasitic capacitances also appear in the set of parasitic capacitances for modules (CR_M). This implies that when generating the parasitic-inclusive topology, these capacitances can be ignored because they are already present in the modified module-topologies. The interconnect parasitic capacitances that do not appear in the set of module parasitic-capacitances have to be added to the topology being generated as new modules themselves. The set of such capacitances ($C_{IC,Add}$) can be found thus:

$$C_{IC,Add} = C_{IC} - (CR_M \cap C_{IC}) = \{C_{1,2}\}$$

Each capacitance in this set becomes a new module in the parasitic-inclusive topology.

The various parts of the parasitic-inclusive topology have been generated. The individual parts (modified module-topologies and capacitance modules) are stitched together according to the connection template of the original circuit topology to produce the parasitic inclusive topology. Figure 3(A) presents the topology generated after inclusion of the parasitic capacitances, which will be used for hierarchical symbolic analysis. The capacitance $C_{1,2}$ is the only capacitance module (Cap) added to the original circuit topology. The topology of modules, which had been selected based on the original circuit topology, have been modified to eliminate some capacitances. The capacitances retained for each module are listed: CM ($C_{4,3}, C_{4,0}, C_{3,0}$); DP ($C_{1,5}, C_{5,3}, C_{2,4}, C_{4,5}, C_{1,0}, C_{2,0}, C_{5,0}, C_{1,3}, C_{2,5}$); ST (None).

If the parasitic resistances of interconnects are also extracted then the technique to include them in the new topology differs from that described above. For each net in the layout, a resistance-model (based on the number of terminals in the net) as described in [5], is included in the parasitic-inclusive topology, along with the modified module-topologies. For the circuit under consideration, the parasitic-inclusive topology is shown in Figure 3(B).

2.2 Layout Sampling

In layout sampling technique, we generate a number of sample layouts of the entire circuit, examine which area and coupling capaci-

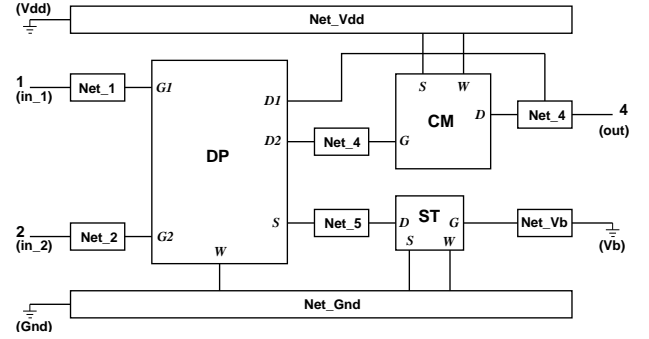
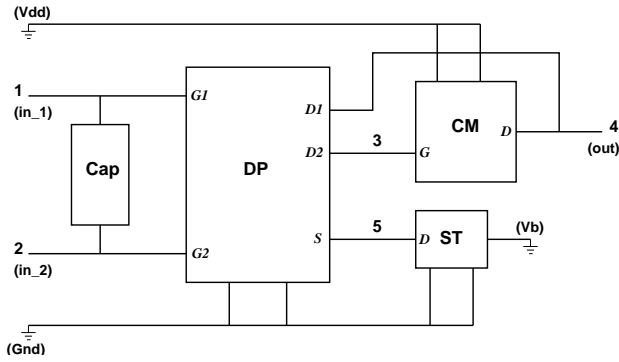


Figure 3: Block Diagrams of Parasitic-Inclusive Circuits (A) Capacitance Included (B) Resistance Included

tances are extracted from these layouts and symbolically include only those capacitances in the SPM generation process. The advantage of using this method is that only relevant parasitics, i.e., those which have appeared due to extraction of layouts, are taken into account.

3. CIRCUIT PARTITIONING

The parasitic-inclusive circuit can be partitioned using the techniques: Module-based partitioning; and algorithm-based/manual partitioning. The aforementioned techniques are described below.

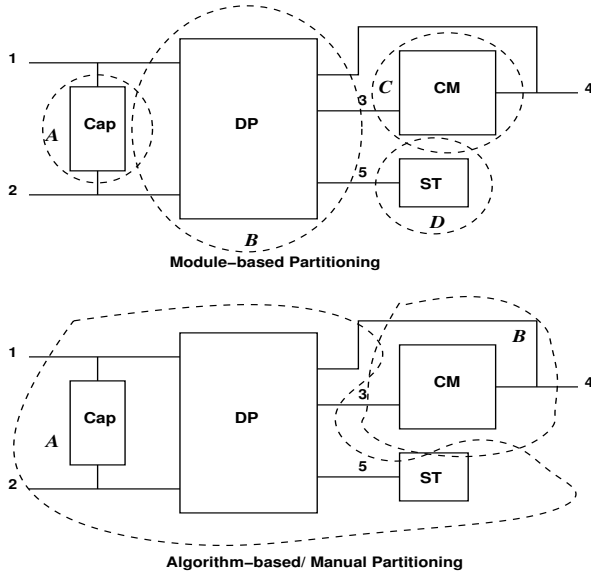


Figure 4: Partitioning of Parasitic Capacitance-Inclusive Circuit

3.1 Module-Based Partitioning

In this approach, the parasitic-inclusive topology is partitioned in accordance with the modules used for its generation. Figure 4 demonstrates such a partitioning, where the topology has been partitioned into four modules: DP , CM , CS and Cap . The advantage of this approach is that, a library of pre-generated transfer functions, in compiled ECD form, can be used to build up the desired transfer function.

3.2 Algorithm-Based/ Manual Approach

The alternative to module-based partitioning is a partitioning algorithm or a manual technique. Instead of partitioning the circuit into several small sub-circuits, breaking the circuit into fewer sub-blocks, which are reasonably large, often lessens the complications involved

in generating the formula for obtaining the desired transfer function. The main disadvantage of this technique is that the transfer functions (i.e., the ECD's) have to be generated for benchmark circuit, and this increases the setup time. In Figure 4 it can be observed that, the number of sub-blocks in algorithm-based/manual partitioning is fewer than the module-based approach. Even though the sub-blocks are larger than modules, they can easily be symbolically analyzed using the ECD Tool.

4. HIERARCHICAL SYMBOLIC ANALYSIS

The method to synthesize the transfer function of a circuit in term of it's subcircuits transfer functions is described here. The key concept introduced in this section is that of a general interconnection template of two subcircuits and the derivation of circuit function matrices.

4.1 General Interconnection Template (GIT)

When two subcircuits are merged the primary goal is to find the circuit function matrix of the new subcircuit/circuit in terms of circuit functions of the constituting blocks. Two subcircuits can be connected to each other in various connection templates whether it be series, parallel, feedback or any combination of these. We start with a connection template which encompasses any kind of connections possible between two subcircuits and carry out our analysis on it. The results obtained from the analysis of the general interconnection template (Figure 5(A)) can be used for analyzing any type of connection by simply eliminating the variables which do not exist in the connection being analyzed. Each subcircuit can be characterized in terms of circuit functions [1] as shown below:

$$\mathbf{M}_x = \mathbf{H}_x \times \mathbf{N}_x \quad (1)$$

The matrix \mathbf{M}_x is the set of primary variables characterizing a port and \mathbf{N}_x is the set of secondary variables. Subscript x represents the subcircuit x which is being characterized. \mathbf{H}_x is the *circuit function matrix*, members of which are obtained by symbolic analysis of the two subcircuits being merged.

From Figure 5(A) the characteristic equations for subcircuit **A** and **B** by using equation 1. All the circuit functions in the matrix \mathbf{H}_x are obtained either by repeated symbolic analysis according to the definition of the circuit functions or have been generated by merging two subcircuits. Similarly, the characteristic equations for subcircuit **B** are obtained.

The key to the synthesis of circuit functions of the new subcircuit/circuit **C** (which is obtained by merging **A** and **B**) is to devise an efficient mathematical procedure of deriving the circuit function matrix \mathbf{H}_c . The characteristic equations of subcircuit **C** are represented as:

$$\mathbf{M}_c = \mathbf{H}_c \times \mathbf{N}_c$$

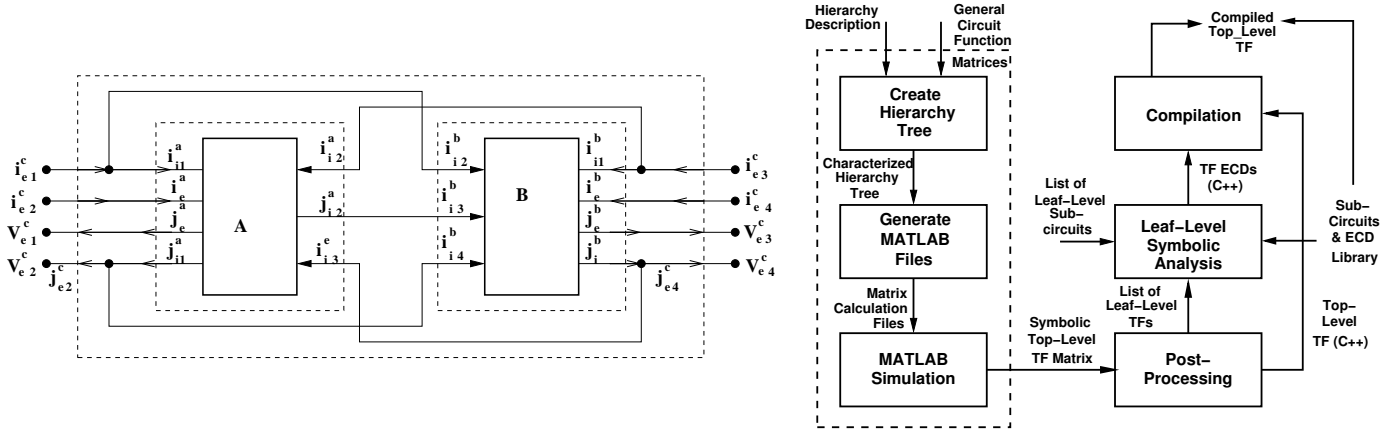


Figure 5: (A) General Interconnection Template (B) Implementation Framework

4.2 Transfer-Function Synthesis of the GIT

In the general interconnection template it is assumed that the sub-circuit **C** is characterized only by external variables. Therefore the compact representation is:

From Figure 5(A) based on KCL and KVL we derive the following relations: $v_{i1}^a = v_{e1}^c$; $v_{i2}^a = v_{e3}^c$; $v_{i2}^a = v_{e4}^c$; $v_{e1}^c = v_{e2}^c$; $j_{i1}^a = j_{e2}^c + i_{i4}^b$; $j_{i2}^a = i_{i3}^b$; $j_e^a = j_{e1}^c$. Also $v_{i1}^b = v_{e3}^c$; $v_{i2}^b = v_{e1}^c$; $v_{i3}^b = v_{i2}^a$; $v_{i4}^b = v_{e2}^c$; $v_e^b = v_{e4}^c$; $j_i^b = j_{e4}^c + i_{i3}^a$; $j_e^b = j_{e3}^c$.

In order to obtain \mathbf{H}_c , the primary internal variables of both subcircuits **A** and **B** have to be eliminated. The first step in that direction is the separation of the circuit functions associated with internal variables into a separate matrix \mathbf{A}_{ab} . The equation resulting is shown below.

$$\begin{aligned} \mathbf{A}_{ab} \times \mathbf{P}_{ab} &= \mathbf{B}_{ab} \times \mathbf{M}_c + \mathbf{C}_{ab} \times \mathbf{N}_c \\ \Rightarrow \mathbf{P}_{ab} &= \mathbf{A}_{ab}^{-1} \times (\mathbf{B}_{ab} \times \mathbf{M}_c + \mathbf{C}_{ab} \times \mathbf{N}_c) \end{aligned} \quad (2)$$

The characteristic equations of the new subcircuit **C** can be formed by substituting the primary external variables of **C** by internal and external variables of the constituting subcircuits based obviously on KCL and KVL. The equation resulting from this

$$\Rightarrow \mathbf{M}_c = \mathbf{D}_{ab} \times \mathbf{M}_c + \mathbf{E}_{ab} \times \mathbf{N}_c + \mathbf{F}_{ab} \times \mathbf{P}_{ab} \quad (3)$$

As can be seen equation 3 still has the internal variable matrix \mathbf{P}_{ab} . In order to eliminate this and get equations in terms of external variables (and hence current and voltage variables of subcircuit **C**), we plug in equation 2 in the equation 3. This results in the following equation:

$$\begin{aligned} \mathbf{M}_c &= \mathbf{D}_{ab} \times \mathbf{M}_c + \mathbf{E}_{ab} \times \mathbf{N}_c + \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times (\mathbf{B}_{ab} \times \mathbf{M}_c + \mathbf{C}_{ab} \times \mathbf{N}_c) \\ \Rightarrow (\mathbf{I} - \mathbf{D}_{ab} - \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{B}_{ab}) \times \mathbf{M}_c &= (\mathbf{E}_{ab} + \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{C}_{ab}) \times \mathbf{N}_c \\ \Rightarrow \mathbf{M}_c &= [(\mathbf{I} - \mathbf{D}_{ab} - \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{B}_{ab})^{-1} \times (\mathbf{E}_{ab} + \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{C}_{ab})] \times \mathbf{N}_c \\ \therefore \mathbf{H}_c &= (\mathbf{I} - \mathbf{D}_{ab} - \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{B}_{ab})^{-1} \times (\mathbf{E}_{ab} + \mathbf{F}_{ab} \times \mathbf{A}_{ab}^{-1} \times \mathbf{C}_{ab}) \end{aligned}$$

Thus we have a formal method of obtaining the circuit function matrix of the new subcircuit/circuit based in symbolic matrix calculations. This formal method is easy to implement and forms the basis of a hierarchical symbolic analysis method a bottom-up approach is used to combine subcircuits till the highest level of hierarchy is reached. The

following subsection show an example of applying this method to two circuits: one a very loosely coupled circuit (RC Ladder) and the other a maze circuit which is extremely tightly coupled where each node is connected to the other.

4.3 Implementation Framework

We have used the method described in Section 4.2 to develop a framework for hierarchical analysis of large analog networks and practical circuits. The concept is to merge two sub-circuits at a time, till all the subcircuits have been merged to obtain the circuit (transfer) function matrix of the target netlist, in terms of the transfer functions of the leaf-level subcircuits. Figure 5(B) presents the implementation framework of the proposed approach.

The process of hierarchical symbolic analysis takes in as input a hierarchy-description file along with the circuit function matrices of the general interconnection template. A hierarchy description file is the breakdown of the target circuit, in terms of its modular components. These components, in case of analog circuits, can be standard modules like current mirror, differential pair etc, or they can be simply the target circuit partitioned according to an algorithm or manually. This file also has the information about the sequence in which two sub-circuits are combined at a time, as well as the information about nodes and variables, in a format corresponding to the general interconnection template.

The first step is the generation of a hierarchy tree from the hierarchy-description file. Each node in this tree represents a subcircuit, and using the hierarchy description files information about its type and external nodes are also saved. From the general circuit-function matrices, the subcircuit-function matrices are obtained by eliminating the absent variables. The next step is to read the hierarchy tree and generate the Matlab code for the matrix mathematics involved in the process. For the purpose of symbolic matrix multiplication and inversion we have used the Symbolic Math Toolbox in Matlab.

Matlab is invoked to perform the symbolic matrix manipulations. The outputs is a text file with the \mathbf{H}_c of the target circuit or a part of it, depending on the desired transfer function. This output, which is a list of symbolic expressions in terms of leaf-level transfer functions, is then processed to convert it to C++ code. This code becomes the top level function which combines the transfer functions of leaf level circuits (also stored as C++ code), to give the top-level transfer function. The output files from Matlab are also parsed to obtain the list of leaf-level transfer functions required by the top-level transfer function(s).

The list of transfer functions and subcircuits are fed to a symbolic analyzer. The symbolic analyzer used in our work is an ECD-based symbolic analysis tool [7]. The ECDs are generated as C++ code

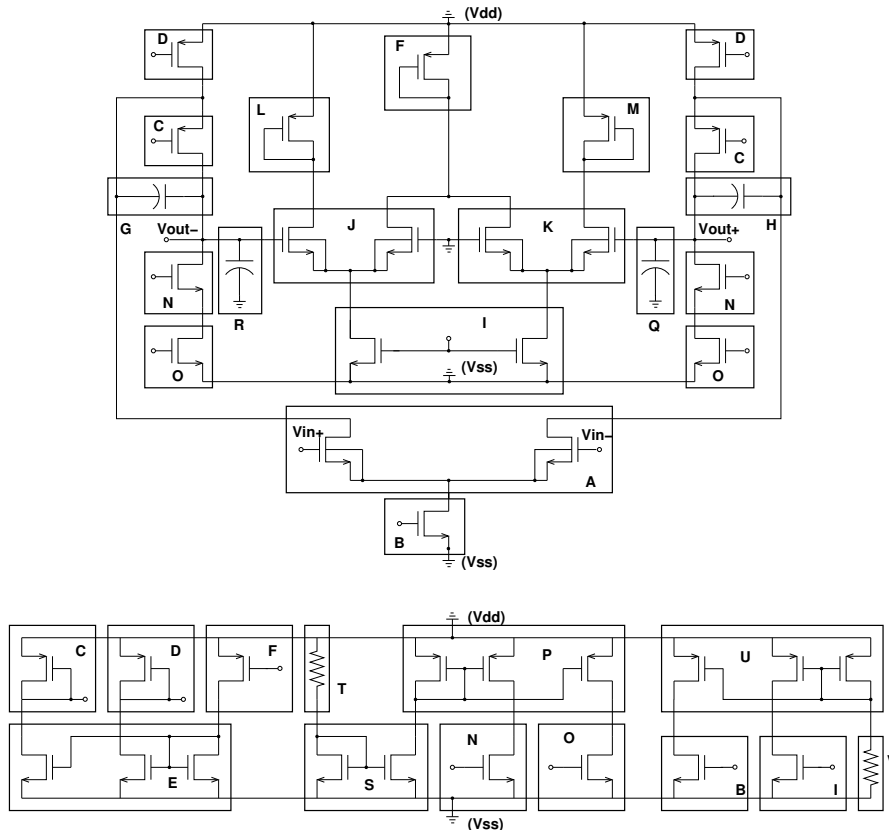


Figure 6: Differential Operational Amplifier

Table 1: Results of Flat Symbolic Analysis of Large Practical Analog Circuits

Circuit	ECD Generation		ECD Statistics			Lines of C++ Code	ECD Compilation Time
	Time	Memory	Depth	# Vertices	# Edges		
FDO_NO-PAR	3m 36s	210M	27	516,905	3,888,252	517,502	C/F
FDO_PAR	7m 5.2s	663M	27	683,839	7,707,532	684,645	C/F

which can be compiled and stored for extremely fast-evaluation. It may not be necessary to generate all or any of the ECDs, if the circuit is partitioned into common-analog modules. Most of the common modules can be analyzed completely and stored as a library of pre-compiled ECDs. This method can be especially useful in very large circuits, generally parasitic-inclusive circuits.

5. SYMBOLIC PERFORMANCE MODELING

The framework for the generation of SPMs was first proposed in [5]. The first step is to generate combinations of nodes that appear in a performance characteristic formula. All the active devices in a circuit are expanded to their small-signal models. The symbolic analysis engine uses the node information to generate the required transfer functions as Element Coefficient Diagrams (ECDs). The symbolic model builder uses the node information to generate the formulae for the desired performance characteristics. The combination of the symbolic formulae and transfer functions are called SPMs.

6. SYNTHESIS FRAMEWORK

The proposed circuit synthesis environment was first proposed in [5]. Layouts are generated by using the Module Specification Language(MSL) system [4], which produces parameterized layouts. A

parameterized layout is a fixed template layout, which when provided with the values of the circuit *parameters* by the optimization engine, produces a physical layout. In our case simulated annealing is used for optimization. A standard circuit extractor is used to extract the devices and parasitics from the layout. The extracted parasitic values along with the passive component values are passed to the pre-compiled SPMs. The SPMs also take in the small-signal parameter values for all active devices obtained by performing an operating point analysis using SPICE. The performance estimates obtained from SPMs are compared to the specified constraints. If necessary, the optimization engine proposes a new set of design parameter and this process continues till convergence.

7. EXPERIMENTAL RESULTS

The results of flat (direct) symbolic analysis of these analog circuits are presented in Table 1. Table 2 presents the results of symbolic analysis based on synthesis of transfer functions. The first observation to make is that that the symbolic matrix calculations in Matlab are extremely efficient. This is obtained by spitting each level of transfer function synthesis into different Matlab files and evaluating all of them in the same session. Matlab can become extremely inefficient if all the symbolic matrix manipulations are specified in just one file. The large analog circuit chosen for our experiments is a fully differential

Table 2: Results of Hierarchical Symbolic Analysis of Large Practical Analog Circuits

Circuit	Transfer Function Synthesis Time					ECDs of Leaf-Level Subcircuits		
	Matlab Files Generation	Matlab Simulation	Top Level C++ File	Compilation of C++ file	Total Time	# of ECDs	Generation Time	Compilation Time
FDO_NO-PAR	3.9s	18.3s	149s	23m 5s	25m 34s	166	22m 24.6s	15m 47.9s
FDO_PAR	3.9s	18.3s	149s	23m 5s	25m 34s	166	22m 42.8s	16m 10.1s

Table 3: Layout-Inclusive Synthesis Results: FDO_PAR - Run #2

Parameter	Specs	Flow1	Flow2	Verif.	%Err
$gain$ (dB)	≥ 45	45.2	46.7	46.5	0.4%
f_{-3dB} (kHz)	≥ 500	711.8	609.3	615.1	1%
ugf (MHz)	≥ 5	12.8	13	13.4	3.1%
pm ($^{\circ}$)	≥ 80	86.3	86.3	84.2	2.4%

Table 4: Time Results of Layout-Inclusive Synthesis for FDO_PAR

Run	Synthesis Approach & Speedup	Time per Iteration				Total Iter. Time
		Layout Gen. & Ext.	Performance Estimation			
			Numer. Analysis	Model Eval.	Total	
# 1	Flow1	1.491s	0.512s	0.372s	0.885s	2.379s
	Flow2	1.491s	0.277s	0.011s	0.29s	1.797s
	Speedup	-	1.85x	33.8x	3.05x	1.32x
# 2	Flow1	1.491s	0.895s	1.002s	1.9s	3.395s
	Flow2	1.491s	0.277s	0.011s	0.29s	1.797s
	Speedup	-	3.23x	91.1x	6.55x	1.89x

opamp (FDO) shown in Figure 6. In the figure we have also shown the module-based partitioning with rectangular boxes with solid lines.

We observe in Table 1 that for FDO_NO-PAR (module-based partitioning, no parasitics), flat symbolic analysis can generate the ECD very quickly, but due to the large size of the ECDs the generated C++ code is huge. This leads to compilation failure (C/F). Thus we know that flat symbolic analysis is inadequate for fast performance evaluation in each synthesis iteration. A similar trend is observed for FDO_PAR (module-based partitioning, module parasitic capacitances). In Table 2, we observe that for both cases, the hierarchical method can generate and compile the ECD and top level C++ files. In the case of FDO_NO-PAR the transfer function generation time is similar in both the flat and hierarchical cases, if ECD-libraries are used. In smaller circuits the flat approach generates faster than the hierarchical approach. As the size of the size of circuits increase hierarchical analysis becomes faster relative to the flat approach. This is seen in the case of FDO_PAR where the time taken by flat symbolic analysis to generate the transfer functions is almost thrice the time for the hierarchical method. If ECDs have to be generated then hierarchical approach an additional setup time.

Table 3 presents the results for two approaches of circuit synthesis. The first approach (Flow1) uses the MSL environment and NG-Spice for layout-inclusive synthesis. The second approach (Flow2) uses MSL with hierarchical SPMs. The inclusion technique used is layout-template analysis, but only the module parasitic capacitances are included. The layout editor used in both approaches is Magic 7.1 and the numerical simulator is Spice. The obtained performance estimates for Flow2 are verified using NG-Spice. The verification is done not against a complete extracted netlist for the final design point, but with a extracted netlist with only the module capacitances. The small error can be attributed in part to the bisection method of solving for roots of the transfer function, to obtain the values of unity gain and

-3dB frequencies. Table 4 presents the time results for both methodologies. The performance convergence in both cases is fast due to the use of a language-based layout generator. The speedup depends on the number of ac analysis points used in numerical simulation with Spice. Run1 uses 10 points/decade and Run2 uses 25points/decade. Flow2 is 1.3x & 1.9x faster than Flow1, per iteration for Run1 and Run2 respectively. The numeric analysis time of Flow2 (for operating point analysis) is 1.9x & 3.2x faster than Flow1. Model evaluation time of Flow2 (SPM evaluation time) is 33x & 91x faster than model evaluation (the process of obtaining desired performance attributes from the analysis results) time for Flow1. Overall, SPM-based performance estimation time is 3x & 6.6x faster than NG-Spice-based method. All experiments have been conducted on SunBlade 1000 with Solaris(SunOS), 2048MB RAM and 2-750MHz Processors.

8. CONCLUSIONS

In this paper we have presented a method for parasitic-aware hierarchical symbolic performance modeling. The underlying technique is the transfer-function based hierarchical symbolic analysis technique. The efficiency of the hierarchical symbolic analysis has been demonstrated. We have also developed techniques to generate the parasitic-inclusive topology directly from the layout template. A layout sampling technique can also be used for generation of the topology. The transfer-function synthesis based hierarchical symbolic analysis paves the way for library-based symbolic analysis techniques, where pre-generated ECDs of common analog modules (like current mirrors, differential pairs etc.) are used to obtain the circuit function of large analog circuits.

9. REFERENCES

- [1] S. Djordjevic and P. Petkovic. A hierarchical approach to large circuit symbolic simulation. *Microelectronics Reliability, Elsevier Science Ltd*, 41:2041–2049, 2001.
- [2] A. Doholi and R. Vemuri. A regularity-based hierarchical symbolic analysis method for large-scale analog networks. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(11):1054–1068, Nov. 2001.
- [3] G. Gielen and R. Rutenbar. Computer-aided design of analog and mixed-signal integrated circuits. *Proc. IEEE*, 88(12):1825–1854, Dec. 2000.
- [4] H. Sampath and R. Vemuri. MSL: A high-level language for parameterized analog and mixed signal layout generators. In *12th Intl. IFIP VLSI Conf.*, 2003.
- [5] M.Ranjan et al. "Fast, Layout-Inclusive Analog Circuit Synthesis using pre-compiled parasitic-aware symbolic performance models". In *Proc. of DATE*, pages 604–609, Feb 2004.
- [6] X.-D. Tan and C.-J. R. Shi. Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(4):402–412, Apr. 2000.
- [7] W. Verhaegen and G. Gielen. "Efficient DDD-based Symbolic Analysis of Large Linear Analog Circuits". In *Proc. of DAC*, pages 139–144, 2001.