SYMBOLIC EXTRACTION FOR ESTIMATING ANALOG LAYOUT PARASITICS IN LAYOUT-AWARE SYNTHESIS

I. TSENG[†] , A. POSTULA[†] , AND L. JÓŹWIAK[‡]

[†] THE UNIVERSITY OF QUEENSLAND, AUSTRALIA
[‡] EINDHOVEN UNIVERSITY OF TECHNOLOGY, THE NETHERLANDS

KEYWORDS: Symbolic Extraction, Analog Design Methodology, Layout Parasitic Extraction, Physical Design, Parameterized Layout

ABSTRACT: This paper presents a new layout parasitic extraction paradigm, symbolic extraction, for use in layout-aware analog synthesis methodologies. Unlike traditional post-layout extraction, symbolic extraction extracts layout parasitics in symbolic form from parameterized layouts. As a result, parasitic values can be calculated directly from given circuit and layout parameters. In layout-aware circuit synthesis process, tasks of time-consuming layout re-generation and re-extraction can be replaced by this fast parasitics calculation step. In the paper, we discuss how to integrate symbolic extraction into the existing analog design flow and how symbolic extraction can be implemented.

INTRODUCTION

One critical problem in analog integrated circuit design is layout-induced parasitics. Layout parasitics can have significant effects on design performance, especially for high-frequency circuits. In an experiment for impact of parasitics [1], design performance can be shifted as high as 90% if parasitics are not considered.

In order to take layout-induced effects into account in analog design process and automate the redesign loops, layout-in-the-loop synthesis methodologies [2, 3] have been proposed, as illustrated in Fig. 1(a). In these methodologies, layout generation and parasitic extraction steps are required in that we need to know the performance degradation induced by the layout. The results of extracted net-lists are then used to guide synthesis tools for the next circuit sizing loop. Although layout generation and extraction steps are very important to the methodologies, they are very time-consuming.

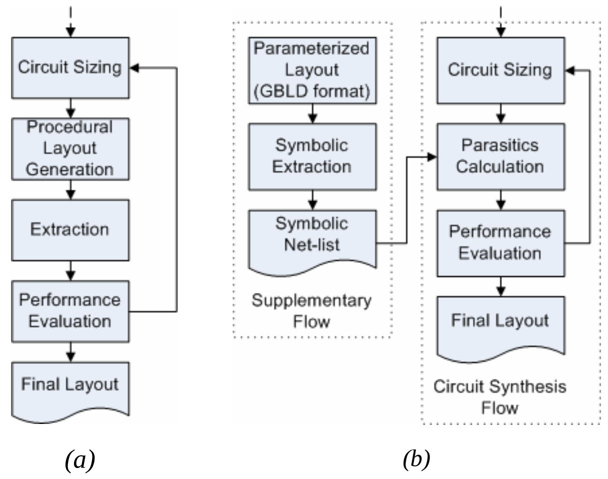


Fig. 1. Analog design methodologies: (a) Layout-in-the-loop (Layout-inclusive) approach (b) Proposed approach

Novel methodologies toward solving slow layout generation and extraction in layout-aware synthesis flow have been proposed. In [1, 4], the approach of using Module Characterization Table (or MCT, a type of look-up tables) to estimate parasitic values has been proposed. However, one problem of this approach is that the size of an MCT can grow exponentially, especially when there are many variables for the input column. Another problem is that the routing is oversimplified. The use of routing boxes, that each of them consists of a horizontal and several vertical segments, is not the case in general, especially for high-performance analog circuits. Proposed by the same research group at the University of Cincinnati, the pre-layout extraction approach uses a high-level language, MSL, to generate extracted net-lists from given circuit parameters without generating a concrete layout [5]. However, there are disadvantages for this methodology. One drawback is that it is not simple to define extraction sections in MSL. To specify rules and variables in an extraction section, designers first have to analyze the module generator and have to have a picture in mind in advance for what the final extracted net-list will look like. Then they have to manually define the connections of each active and passive element by using the language. After that, designers assign the associated variables or values to those elements. The procedures above are like coding parasitic extractors manually for each module. In addition, each module generator must have at least one version of its own extraction sections for the purpose of different levels of accuracy. For large modules, therefore, using the language to define the extraction sections would be a very tedious work.

Our idea is similar to pre-layout extraction and the MCT approach because no detailed layout is generated in order to attain extracted net-lists during layout-inclusive synthesis process. On the other hand, our approach has the following distinctions. (1) Extracted net-lists in our approach are in symbolic form. For instance, the resistance value for a parasitic resistor is an equation. Exact values can be computed rapidly from the given parameters. (2) The extracted net-lists depend on the values of circuit parameters and layout parameters, not just circuit parameters. That dramatically shortens the circuit synthesis loops [6]. (3) Designers do not have to code the extracted circuits as using the MSL language. The net-list structure is generated automatically. (4) The extraction process is carried out only once. In layout-aware synthesis process, parasitics are calculated rather than extracted.

In our design flow, as shown in Fig. 1(b), designers have to design their layout generators for each module, as using CAIRO [7] or other module generation languages in the layout-in-the-loop flow. The language that we use in our design flow is GBLD [8]. After performing symbolic extraction on the GBLD code, a net-list with parameterized parasitic values and/or conditional statements is generated. We call the generated net-list *symbolic net-list*. In layout-inclusive circuit synthesis loops, exact parasitic values can be calculated and the extracted net-list can be attained rapidly from the symbolic net-lists. In our approach, expensive layout generation and traditional parasitic extraction steps are not needed in the synthesis flow.

The rest of this paper is organized as follows. In the next section, we introduce our parameterized layout format, GBLD, which can also be used as the language for designing module generators. Then we describe how to perform symbolic extraction from parameterized layouts. Concluding remarks and future work are in the final section.

TABLE 1. GBLD Symbols

Symbols	Actions/Meanings
F	Move forward one unit in the current direction and draw a line
f	Move forward one unit in the current direction (without drawing a line)
+	Turn left 90 degrees
–	Turn right 90 degrees
{	Start recording a string except “{”, “}”, and “!” on a magnetic tape
}	Stop recording and place an end-of-the-record symbol on the tape
!	Rewind the magnetic tape to the previous end-of-the-record symbol, generate and erase the recorded characters of the string between the two end-of-the-record symbols, then move to the end of the recording
F(***n***)	Repeat the “F” symbol exactly ***n*** times
f(***n***)	Repeat the “f” symbol exactly ***n*** times
<***nt***,“***ly***”>	Generate the string “***ly***(***p***)” if there is a production rule defined as “<***nt***>→***p***” and <***nt***,“***ly***”> can only appear on the right hand side of production rules. (***ly*** stands for layer name or layer number. The symbol ***p*** stands for a polygon or polygons.)

PARAMETERIZED LAYOUT FORMAT

Grammar Based Layout Description (GBLD) is our parameterized layout format [8]. The format can be used to represent geometrical layouts as well as parameterized layouts. Most of the symbols defined in GBLD are borrowed from L-systems [9] and context-free grammars. In Table 1, we re-list some of the GBLD symbols which are used in this paper for the reason of completeness. For other unlisted symbols and related examples, readers can refer to another of our published papers [8].

GBLD can be considered as a limited version of procedural module generators, but there exists benefits for the limited description power. One advantage is that we can extract symbolic net-lists easier from GBLD than from other languages which have more complicated syntax, such as loops and conditional branches. In addition, we can always define new symbols into GBLD so that the description power can be improved. Nevertheless, GBLD can be used to represent parameterized layouts for the following important aspects: device dimensions, relative positions, routing controls, and transformations for wires.

Device Dimensions

For procedural module generators used in analog design, the capability of generating layouts for different transistor lengths and widths is necessary. This is because transistor lengths and widths (or the W/L ratios) are important parameters that very much influence the circuit performance. During circuit sizing process, tuning those parameters is one of the major tasks. GBLD is able to represent parameterized layouts of transistors by keeping the channel lengths and widths as parameters.

In Fig. 2, for example, there are two rectangular polygons. One is on the n-diffusion layer, and the other is on the polysilicon layer. Note that the small black dot in the figure is the starting point and the default direction is upward. Fig. 3 shows the GBLD code for describing the parameterized layout in Fig. 2. The code is very concise since we use an imaginary magnetic tape to describe the repetitive portions. In the description code, the non-terminal symbols <length> and <width> can be treated as parameters. If we assign actual values to the parameters, such as <length>→2 and <width>→5, the detailed layout can be represented and generated.

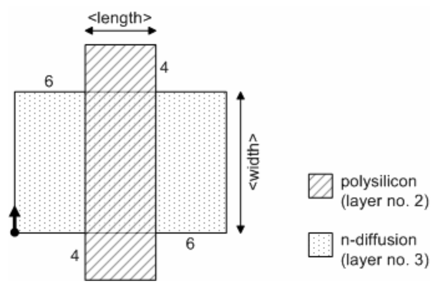


Fig. 2. NMOS transistor with parameterized channel length and channel width

```
<NMOS> → <Ndiff, "3"> – f(6) – f(4) – – <Poly, "2">
<Ndiff>  → { F(<width>) – F(6) F(<length>) F(6) – } { ! }
<Poly>   → { F(4) F(<width>) F(4) – F(<length>) – } { ! }
```

Fig. 3. The GBLD code for the MOS transistor with parameterized channel length and channel width

Relative Positions

For the layout-aware analog synthesis approach proposed in [10], defining relative positions is an important task. By defining parameters to the relative positions of blocks, placement is simplified and the complexity of synthesis procedures is reduced. GBLD can be used to represent relative positions of analog layout blocks. As shown in Fig. 4, there are two layout blocks, A and B. The relative position of A and B can be defined via the positions of point *p* and *q*, as well as via any other point in block A and any other point in block B. The pseudo GBLD code for representing the relative position in the figure is:

> *<Block A ends at point p with upward direction>* **– <x> + <y>** *<Block B starts at point q with upward direction>*

The parameters **<x>** and **<y>** in the code represent the horizontal and vertical distances between points *p* and *q* respectively, as we can see in the figure.

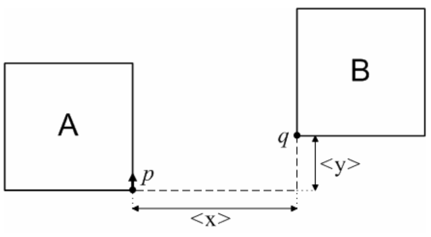


Fig. 4. Relative position of two layout blocks

Routing Controls

In Fig. 5(a), there are two layout blocks A and B. The wire between A and B connects the two blocks. The vertical part in the middle of the wire is stretchable. The vertical part is <Param> units in length and can also go down or go straight as in (b) and (c) respectively. GBLD can describe these cases in the code below.

```
<Wire>     → F – F(4) { <Turn> } F(4) – F – F(4) { ! } F(4)
<Turn>     → <Up> | <Down> | <Straight>
<Up>       → + F(<Param>) – F
<Down>     → F – F(<Param>) +
<Straight> → F
```

We can also treat the two non-terminal symbols <Turn> and <Param> as parameters, where <Param> is a real number and <Turn> has only three choices, <Up>, <Down>, and <Straight>. By controlling the two parameters, the wire direction and the wire length can be altered. We can also control the placement of B relative to A as well. Equations of parasitic values for the wire can be derived by having <Turn> and <Param> as parameters.

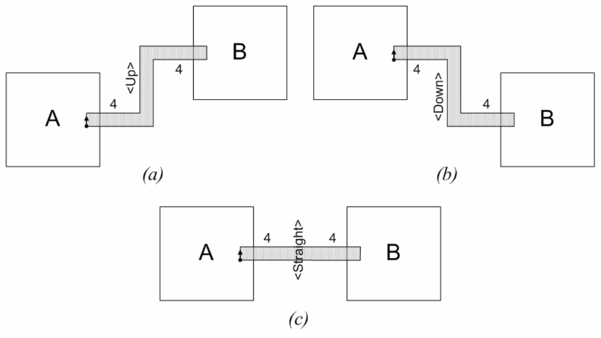
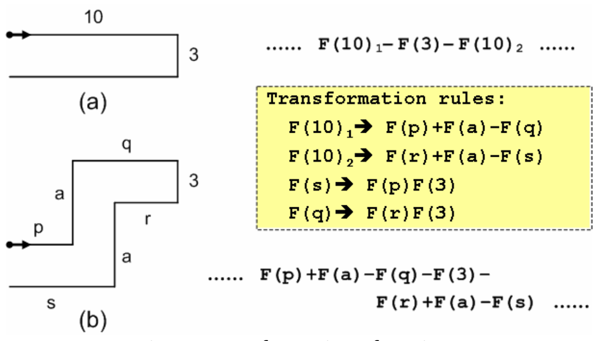


Fig. 5. Routing controlled by parameters

Transformations for Wires

In this subsection, we show how a fixed-coordinate wire can become bendable and stretchable. A portion of a horizontal wire is shown in Fig. 6(a). The wire is 10 units in length and 3 units in width. The corresponding GBLD code for the wire portion is "… $F(10)_1$ – F(3) – $F(10)_2$ …". We use the subscript numbers to differentiate the two F(10)'s before and after F(3). By performing the transformations using the transformation rules shown in the figure, the final GBLD code is "… F(p) + F(a) – F(q) – F(3) – F(r) + F(a) – F(s) …", where a, p, and r are parameters that can be used to control the geometry. The value of s is the addition of the value p and 3 (mathematical addition). We use the statement F(s)→F(p)F(3) to implement this mathematical addition in GBLD. The relationship between q and r can also be expressed in this way. The final layout after transformations is shown in Fig. 6(b).



```
...... F(10)1-F(3)-F(10)2 ......

Transformation rules:
  F(10)1→ F(p)+F(a)-F(q)
  F(10)2→ F(r)+F(a)-F(s)
  F(s)→ F(p)F(3)
  F(q)→ F(r)F(3)

...... F(p)+F(a)-F(q)-F(3)-
                F(r)+F(a)-F(s) ......
```

Fig. 6. Transformations for wires

SYMBOLIC EXTRACTION

In this section, we discuss how to perform parasitic extraction on parameterized layouts which are represented by GBLD. We call the process *symbolic extraction*. For the examples in this section, we model capacitive and resistive parasitics of interconnections. The following three types of parasitic capacitances are considered: (a) area (parallel-plate) capacitance to ground, (b) fringing capacitance (or fringe capacitance) to ground, and (c) inter-wire capacitance between parallel wires placed on the same layer. Values of these three types of capacitances are proportional to wire lengths, and are inversely proportional to separation distances. For parasitic resistances, they are also proportional to wire lengths if we ignore the skin effect. After performing symbolic extraction, therefore, the final results for capacitive and resistive parasitic values are simple mathematical equations, which mainly contain addition, subtraction, multiplication, and/or division. This makes computations of exact parasitic values very fast during layout-inclusive circuit synthesis. We divide symbolic extraction into two types. They are Simple Symbolic Extraction and Conditional Symbolic Extraction. Both of them are presented in the following subsections.

Simple Symbolic Extraction

Fig. 7 shows the layout of two open-ended wires, Wire 1 and Wire 2. The wire widths are W_1 and W_2 respectively. In the figure, both of the two wires are divided into segments and are parameterized. The heights of the wires are assumed to be the same.

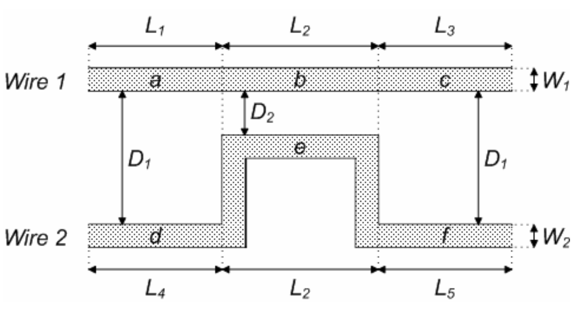


Fig. 7. Symbolic extraction on two parameterized wires

After performing symbolic extraction, the extracted symbolic net-list is illustrated in Fig. 8. For simplicity, we use the T model to approximate distributed RC effects. That means the intrinsic parasitics of a wire segment can be modelled by two serial connected resistors and a capacitor which is connected to the two resistors for one end and to the ground for the other end. For instance, the intrinsic parasitics of the wire segment a in Fig. 7 can be modelled by R_1, R_2, and C_1, as shown in Fig. 8. The capacitor C_4 stands for the coupling capacitance between segment a and segment d.

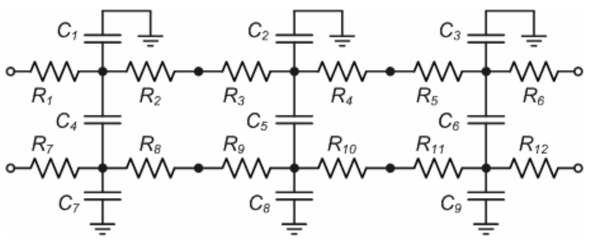


Fig. 8. The extracted net-list of the two parameterized wires

We can start deriving equations for the capacitive and resistive parasitics. For the wire segment a, the area capacitance to ground can be derived as:

$$C_{area} = K_1 L_1 W_1$$

where L_1W_1 is the area of the segment a for the side which is parallel to the ground. K_1 is equal to (ε_{di} / t_{di}), where ε_{di} and t_{di} represent the permittivity of the dielectric layer, SiO_2, and its thickness, individually. Therefore, K_1 is a constant. Note that K_i, for $i \in N$, stands for a constant value in the following equations. The fringe capacitance to ground of the wire segment a can be derived as:

$$C_{fringe} = 2*(L_1K_2)$$

The intrinsic capacitance C_1 is the sum of C_{area} and C_{fringe} , or

$$C_1 = C_{area} + C_{fringe}$$

Similarly, we can also derive the following equations for the parasitics of the line segment a.

$$C_4 = \min(L_1, L_4)*K_3/D_1$$
$$R_1 = R_2 = (L_1*K_4/W_1)/2$$

The coupling capacitance C_4 depends on the minimum value of the length L_1 and L_4, and is reverse proportional to the separation distance D_1. For the intrinsic resistance of the line segment a, it is divided into two resistors, R_1 and R_2. Each of them has half of the lumped value. For other parasitics in the example, they can also be derived in the similar way.

Conditional Symbolic Extraction

In simple symbolic extraction that we discuss in the previous subsection, a parameterized layout must have a fixed layout topology. That means changing parameter values in the parameterized layout will not change the structure of the final extracted symbolic net-list. (Notice that the calculated parasitic values might be changed.)
Conditional symbolic extraction, on the other hand, is different from simple symbolic extraction. In conditional extraction, the final extracted net-lists can have different RC structures which depend on the values of input parameters. For instance, in Fig. 9(a), Wire 1 and Wire 2 are stretchable and the lengths of the two wires can be controlled by the parameters L_1, L_3, and L_4. Similar to the example in the previous subsection, there exists a coupling capacitance between wire segments a and c. However, if the value of L_1 is greater than L_3, the layout topology will become the one shown in Fig. 9(b). Another coupling capacitance, which is between wire segments b and d, will exist.
For the example in Fig. 9, the final extracted symbolic net-list must have conditional branches. That is very much similar to traditional programming languages. The pseudo symbolic net-list in SPICE-like format is:

```
if ( $L_1 > L_3$ ) {
  $C_{cond}$ node-in-seg-b  node-in-seg-d  CVAL
}
```

The capacitor C_{cond} is the coupling capacitor between wire segment b and d. It is generated only when the condition ($L_1 > L_3$) happens. *CVAL* is an equation which takes min(L_2, L_4) and D_2 as parameters ($CVAL = \min(L_2, L_4) * K_5 / D_2$).

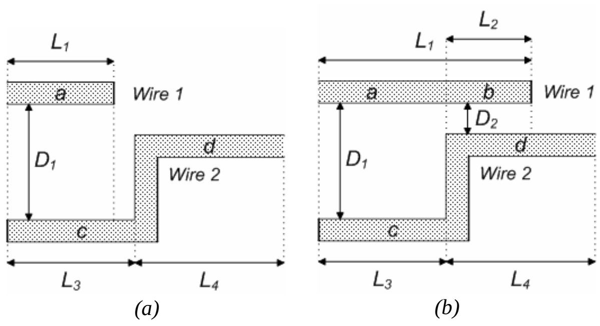


Fig. 9. An example of Conditional Symbolic Extraction

CONCLUSION & FUTURE WORK

This paper presents a new methodology that can replace repetitive time-consuming layout generation and extraction steps in layout-aware circuit synthesis process for analog block level design. Symbolic extraction, which plays an important role in the methodology, is to extract net-lists in symbolic form from parameterized layouts. Exact parasitic values can be computed very rapidly from symbolic net-lists. In our methodology, parameterized layouts are represented by a layout description language called GBLD. For very complex parameterized layouts, the extracted symbolic net-lists may look very similar to programming languages since conditional branches are required.
On going work attempts to build an interactive parameterized layout design system which has symbolic extraction and design-rule-built-in functionalities. Efforts will also be made to tune the design performance based on symbolic net-lists.

THE AUTHORS

I-Lun Tseng and Dr. Adam Postula are with the School of Information Technology & Electrical Engineering, The University of Queensland, St Lucia, Queensland 4072, Australia.
E-mail: {iltseng, adam}@itee.uq.edu.au

Dr. Lech Jóźwiak is with the Department of Information and Communication Systems at the Faculty of Electrical Engineering, Eindhoven University of Technology, The Netherlands.
E-mail: L.Jozwiak@tue.nl

REFERENCES

[1] A. Agarwal, H. Sampath, V. Yelamanchili, and R. Vemuri, "Fast and Accurate Parasitic Capacitance Models for Layout-Aware Synthesis of Analog Circuits," Proc. Design Automation Conference, San Diego, CA, USA, 2004, pp. 145-150.

[2] M. Dessouky, M.-M. Louerat, and J. Porte, "Layout-oriented synthesis of high performance analog circuits," Proc. IEEE Design, Automation and Test in Europe (DATE), Paris, France, 2000, pp. 53-57.

[3] P. Vancorenland, G. V. d. Plas, M. Steyaert, G. Gielen, and W. Sansen, "A layout-aware synthesis methodology for RF circuits," Proc. International Conference on Computer Aided Design (ICCAD), San Jose, California, 2001, pp. 358-362.

[4] A. Agarwal, H. Sampath, V. Yelamanchili, and R. Vemuri, "Accurate Estimation of Parasitic Capacitances in Analog Circuits," Proc. IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE), Paris, France, 2004, pp. 1364-1365.

[5] R. F. Badaoui, H. Sampath, A. Agarwal, and R. Vemuri, "A High Level Language for Pre-Layout Extraction in Parasite-Aware Analog Circuit Synthesis," Proc. Great Lakes Symposium on VLSI (GLSVLSI), Boston, Massachusetts, USA, 2004, pp. 271-276.

[6] I.-L. Tseng and A. Postula, "A Layout-Aware Circuit Sizing Model Using Parametric Analysis," Proc. The 12th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI), Kanazawa, Japan, 2004, pp. 235-240.

[7] M. A. Dessouky, A. Greiner, and M. M. Louerat, "CAIRO: A hierarchical layout language for analog circuits," Proc. Mixed Design of Integrated Circuits and Systems (MIXDES), Krakow, Poland, 1999, pp. 105-110.

[8] I.-L. Tseng and A. Postula, "GBLD: A Formal Model for Layout Description and Generation," Proc. Forum on specification & Design Languages (FDL), Lille, France, 2004, pp. 660-670.

[9] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan, *The algorithmic beauty of plants*. New York; Berlin: Springer-Verlag, 1990.

[10] H. Tang and A. Doboli, "Employing layout-templates for synthesis of analog systems," Proc. IEEE 45th Midwest Symposium on Circuits and Systems (MWSCAS), 2002, pp. II-505-II-508.