This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication. The final version is available at http://dx.doi.org/10.1016/j.vlsi.2018.11.005

A Tool for the Automatic Generation and Analysis of Regular Analog Layout Modules

Ismael Lomelí-Illescas[1,2], Sergio A. Solis-Bustos[1], and José E. Rayas-Sánchez[2]

[1] Intel Corp., Zapopan, 45019 Mexico (ismael.lomeli@intel.com; sergio.solis@intel.com)

[2] Department of Electronics, Systems, and Informatics, ITESO – The Jesuit University of Guadalajara, Tlaquepaque, 45604 Mexico (erayas@iteso.mx)

***Abstract*— This paper describes the characteristics of a new CAD tool that enables the creation of layout libraries of selected analog modules. This Analog Modules Generator (AMG) automatically creates multiple layout versions of two commonly used analog structures: the differential pair and arrays of series-connected or stacked devices, for the subsequent generation of layout libraries. Based on the number of devices and rows defined by the user for the layout implementation, the tool validates all possible implementations, which are later saved in a database. Additionally, an extraction process can be optionally executed over all the layout views saved in the database. The AMG generates several reports with all the characteristics of the implemented layouts, including area and parasitic components, facilitating further statistical processing. We describe the features and capabilities of the proposed AMG tool, and several test cases are presented. Results show that suitable layout implementations can be achieved by layout and circuit designers in a very reduced amount of time.**

***Index Terms*— Analog layout automation, differential pair, layout libraries, layout characterization, stacked devices.**

1. Introduction

SOME of the inherent limitations for the layout implementation of analog devices and structures in current CMOS subnanometric technologies (process technologies below 32 nm) are addressed in [1]-[3]. Among these limitations, the discrete values for the single transistor's diffusion width and fixed values for the transistor's gate length impose constraints for analog design; this has forced the designers to implement new structures to get the correct analog behavior. One solution is the use of stacked devices (transistors connected in series, with a common gate terminal. connecting the source terminal of one transistor with the drain terminal of the next one), as described in [4] and [5], where limitations given by discrete and fixed values are also addressed. In this context, analog layout design is strongly constrained by the transistor's dimensions, connections, and topologies used; therefore, a correct layout implementation becomes critical to achieve the expected performance. To implement and evaluate different layout placements and analyze their tradeoffs while saving time and layout design resources, it would be very desirable to have a CAD tool that enables fast and efficient implementation of analog layouts while providing useful information on their characteristics for its analysis; allowing designers to choose the alternative that best meets their requirements.

A large number of CAD tools have been proposed to automate the generation of analog layouts. The constructive approach [6] implements its placement solution by selecting one module (a transistor or a group of transistors) at a time and placing it in the "best" possible location. The constructive approach has the main drawback of being dependent on the selection order of the modules. On the other hand, the schematic-driven approach [7], [8] can be considered a tool for initial positioning which eases the layout work by generating [8] or comparing [7] a preliminary placement of the devices, considering their position in the input schematic or considering netlist indications. This approach requires significant user-tool interaction.

Another common technique is based on constraints definition [9], used either to define the location and placement of the different devices [10] or to define the routing paths and, in consequence, the location of the different components [11]. In general, most of these approaches require intensive user-tool interaction, as well as significant user's experience in analog layout design.

Other effective layout CAD tools are template-driven. These are built on template databases containing analog circuits designed by experienced designers that guide the generation of the new layout [12]. Another one employs the layout retargeting technique, that consists of generating a new layout from an existing one, typically used in the design migration from one technology process to a new one [13]. Examples of the layout retargeting technique are presented in [14], where foundry Parametric Cells (pCells) are used; in [15], not only layout geometries or building blocks are transferred, but also different constraints from different sources. The retargeting technique allows generating a new layout by acquiring and keeping the design expertise from previous designs. However, it is not very helpful when new layouts with different characteristics are to be implemented, or when different implementations must be compared.

The CAD tool presented in [16] is a layout automation approach based on the concept of cellular automata: each device is modeled as a so-called agent, which autonomously moves, rotates and deforms itself depending on the actions of its neighborhood. This approach leads to a self-organized layout solution. Other related approaches base their placement

I. Lomelí-Illescas is funded through a CONACYT scholarship (*Consejo Nacional de Ciencia y Tecnología*, Mexican Government).

solutions on symmetry constraints [17]-[18]. In those cases, it is necessary to manually create a basic layout configuration and structure/modules, from which the CAD tool can create new ones.

The development of a CAD tool that helps to speed up the layout implementation of basic analog modules or structures focused on differential pairs and arrays of series-connected transistors is described in [1]. That analog module generator (AMG) tool provides different configurations and setting options that allow the users to come up with a suitable layout version. Each layout version is included in a database and used for the generation of an analog layout library. Designers can use the analog modules of this database to integrate them into more complex designs. In this paper, the description of a significantly enhanced layout automation tool is presented. Our new approach is an extension of the work presented in [1]. Some of the main characteristics of both versions of the AMG tool are the following:

a) Layout placement uses an internal database with different topologies defined for the two previously mentioned fundamental analog modules. The database uses placement and routing pattern definition templates to indicate the order and the location in which the devices and the routing metals are drawn. By virtue of these pattern definition templates, the user does not need to write the location for every single device.
b) Users can define which metal layers must be included in the layout.
c) Each transistor finger is automatically drawn individually and then automatically replicated and allocated to create a complete structure, instead of drawing a single large object. Each of these devices has unique parameters and location.
d) All the automatically implemented layouts are short-circuit clean, and design rules check (DRC[1]) compliant, according to the selected technology process. This implies that no re-routing is needed.
e) Due to the modularity and independence of its scripts, our tool can be migrated to other technology processes by internally changing the technology process file and adjusting the set of design rules (captured manually).
f) The scripts that form the proposed AMG are developed using TCL language. AMG is intended to be a complement of commercial layout tools; it can fit into different commercial design suites by replacing the native tool instructions for drawing the devices and geometries.

For this new version of the proposed AMG tool, some additional features are included, namely:

a) It allows the automatic generation of multiple layout versions for both fundamental modules, varying parameters such as the number of transistors (for the case of stacked devices), the finger's width in discrete values defined by the technology process, and the number of fingers of each component, etc., producing a layout database or library.
b) The parasitic extraction is not a separate function; this can be optionally executed over the layouts on the database. This allows the statistical investigation of the parasitics of many different implementation variants.
c) A list of valid and invalid layouts is automatically generated.
d) A summary report that includes parasitic elements information, total area, total number of devices and rows, and layout versus schematic (LVS) test results, is also automatically created.

One of the most important contributions of the proposed AMG tool is that it helps the designers to reduce the time for the characterization and analysis of analog structures when a new project starts, or a technology process is introduced. The information generated by the tool allows designers to make a statistical analysis of the parasitic components of the layouts included in the database, as well as comparisons between different layouts to select the most suitable for their implementations.

The list of invalid layouts helps designers to identify structures that are not physically correct by construction even though they are realizable at the circuit level.

The rest of our paper is organized as follows. In Section 2, a general description of the AMG proposed tool, including its main modules, is presented. Sections 3 to 6 describe the characteristics and functionality of these modules. In Section 7, functional tests to illustrate the AMG's capabilities for automatic layout generation are presented. Finally, in Section 8, conclusions are given, including some possible future research work.

2. AMG General Description

AMG allows the implementation of a single layout (SL-mode) or multiple layout versions of the previously mentioned analog modules with its corresponding layout database (DB-mode). This database facilitates designers to obtain useful information for the optimal implementation of their circuits. This information includes the viability of the layout implementation according to the number of devices and their dimensions, the layout area, and the parasitics associated with the layout.

The flow diagram of the AMG is shown in Fig. 1. It follows that one in [1] but it was modified for new features such as a loop for the creation of multiple layouts, the automated execution of parasitics extraction and LVS flows, and the generation of multiple reports.

The first section is the ENVIRONMENT SETTING, where the specifications for the layout implementation are defined. Here, the user defines if single or multiple implementations will be created. In the SL-mode, users specify a netlist and the names of the transistors that form the structure to be implemented; the layout is created using the information from the netlist. In the DB-mode, the user has to specify the structure to be implemented and the termination criteria which can include the area, the layout width, and the layout height, or

[1] The option to run DRC flow is not available for the user; however, all the layouts created were verified for this flow, to guarantee their quality and the correct functionality of the tool.

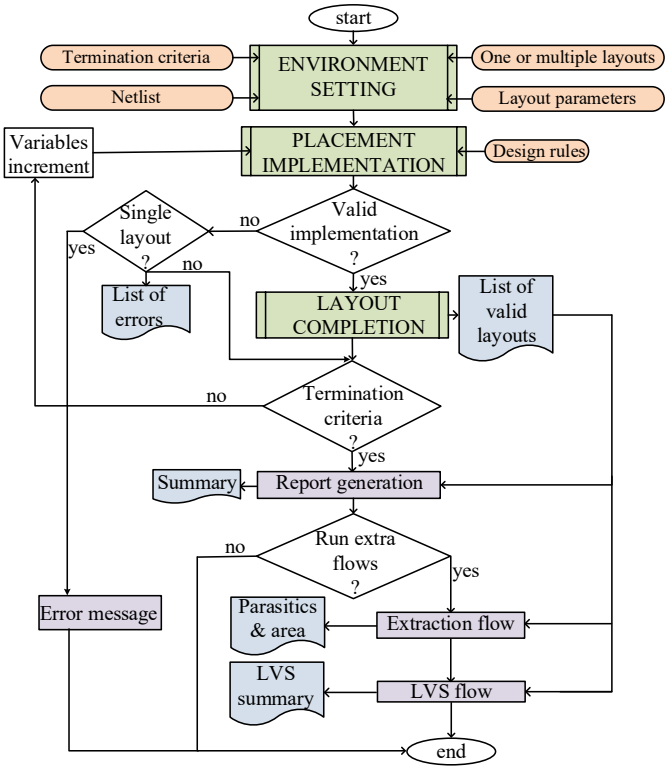

Fig. 1. Flow diagram of the automatic layout generation tool.

the maximum number of valid implementations; the schematic cell and the netlist are created based on the layout to be implemented.

The layout parameters (see Fig. 1) include the transistor model (standard, high speed, low current leakage, low power, etc.), the topology for the layout, the metal layers to be added, the number of fingers and dummy transistors[2], the number of rows, and the insertion of substrate connections.

The second section, PLACEMENT IMPLEMENTATION, performs the placement of the devices; here, the devices are automatically drawn one by one to form the required module. At this stage, only diffusion P or N and polysilicon layers (base layers [1]) are included. Two new files are created: one for a new schematic view and another one for the layout view. An internal function block is responsible for determining the number of rows required for the layout implementation based on the total number of devices, which is updated in each iteration by the "Variables increment" block (see Fig. 1). Then the devices are created in the layout cell.

If a problem is detected during PLACEMENT IMPLEMENTATION, this is registered in an error report file, indicating the reason for this problem. If no errors are found, the LAYOUT COMPLETION section is executed; if a problem is found, this section is skipped, and a report containing the list of errors is generated. If an error is detected for a single layout case, the corresponding error message is displayed, and the AMG stops its execution.

The LAYOUT COMPLETION section in Fig. 1 implements the rest of the layout elements, such as metals, contacts, and special identification layers[3]. Once these layout elements are included, the module is named, and its characteristics are saved in a list of valid layouts. If the completion criteria are satisfied, a report containing all the information related to the implemented layouts is generated, along with a summary file. If the termination criteria are not yet met, then the "Variables increment" block updates the information to start a new placement.

In the AMG, the parasitics extraction is no longer a separate function as it is in [1]. This process can be now executed automatically for all the different layouts implemented and saved in the database. Similarly, the AMG can run an LVS test over all the created layouts.

In the following sections, a more detailed description of the main blocks from the flow diagram in Fig. 1 is presented.

3. Environment Setting

As in [1], in this block, the user-defined specifications and parameters for the single or multiple (library) layout implementation are captured using a graphical user interface (GUI).

The GUI employed in [1] was modified to add the new options. Through this GUI, the user can select the SL-Mode or the DB-mode. In the case of the SL-mode, the user can load a netlist to select the transistors that form the module to be implemented or can select the module to be implemented (differential pair or an array of stacked devices). When a netlist is used, a dedicated function reads it and analyses the connections of the selected transistors, to validate if they form one of the two available modules. From the netlist, the number of fingers, the finger's width and length, as well as the transistor's type and model are obtained. When a netlist is not required, the user has to define those parameters manually. In both cases, only one layout that meets the indicated specifications is created.

In the DB-mode, the user defines the structure to be implemented as well as the termination criteria for the creation of a database. Termination criteria include the maximum number of layout implementations, the maximum layout width, and the maximum layout height. The maximum layout width delimits the maximum number of transistors in a single row, while the maximum layout height delimits the maximum number of rows. The AMG creates all possible layouts that fit into these criteria.

Using the GUI, users can also define if the algorithm of parasitics extraction and the LVS verification flow will be run over the generated database. The rest of the parameters and specifications for the layout implementation are defined either through selection boxes or captured manually. The GUI is shown in Fig. 2.

[2] All terminals of NMOS dummy devices are tied to VSS. Conversely, all terminals of PMOS dummy devices are connected to VDD; the exceptions are the shared drains of the dummy and active transistors.

[3] Special identification layers are placed over the transistors and are used to indicate some specific properties of them, such as low leakage, low power consumption, etc. They are required for the fabrication process and to accomplish the LVS verification flow.

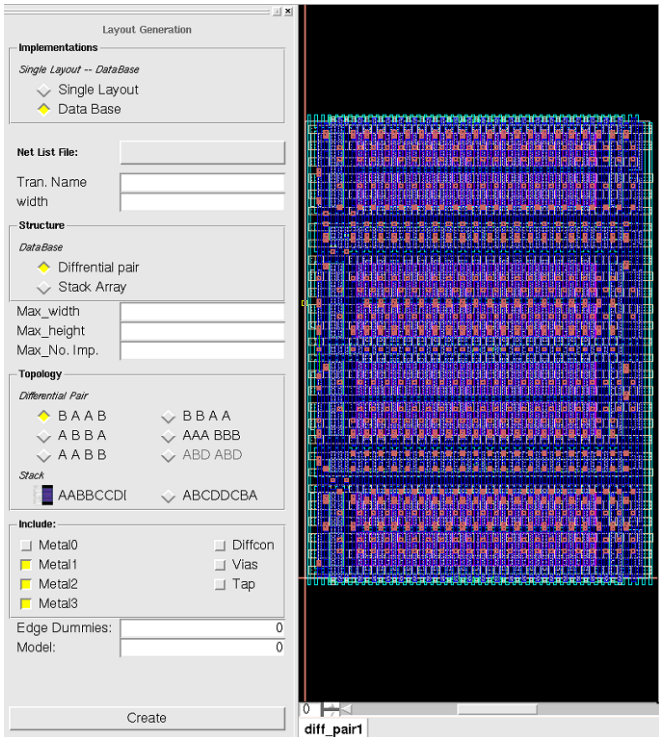

Fig. 2 Sample image of the AMG with a generated layout.

4. Placement Implementation

Once the different settings and layout parameters are defined, the next step is the creation of the required layout versions. In our approach, the PLACEMENT IMPLEMENTATION algorithm, shown in Fig. 3, is executed without interruption until the termination criteria are satisfied. The process to generate the layout is described below.

4.1 Cell Creation and Cleaning

As in [1], two files are created: one for a new schematic view and another one for the layout view. The schematic view includes only the devices and their connections.

4.2 Parameters Modification

The parameters that our AMG tool modifies for the generation of layouts are the number of fingers and their widths. In the case of the stacked devices module, the number of transistors that form the array is also modified. The AMG increases the number of fingers and finger's width, as long as the layout can fit in the available area (termination criteria), using discrete values defined by the technology process; in all the cases, the same transistor's length is used, which is also defined by the technology process.it also distributes the devices into different rows as necessary. The maximum number of fingers and the maximum finger's width value depend on the available area.

Since the AMG modifies the number of fingers and the finger's width, it is possible that different implementations have the same effective width, defined as W_{eff} = transistor's finger width × number of fingers, but a different number of fingers and a different layout placement, this is illustrated in Fig. 4. In Fig. 4a the transistor A is W wide. In Figure 4b the

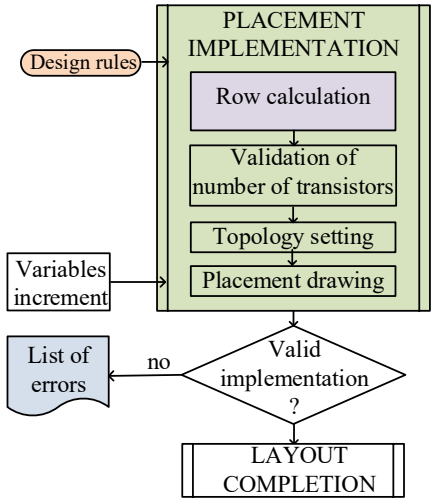

Fig. 3 Flow diagram for the PLACEMENT IMPLEMENTATION section.

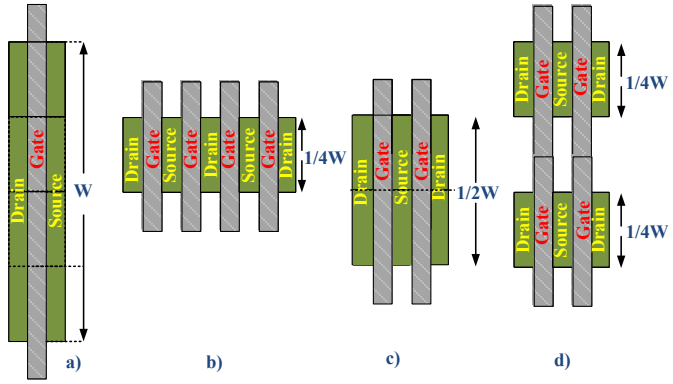

Fig. 4 Example of transistors finger: a) Transistor A is W wide and uses one finger; b) Transistor A is divided into 4 fingers with a width of ¼W; c) Transistor A can be divided into 2 fingers with a width of ½W; d) Transistor A is split into two transistors of two fingers of ¼W width but placed in two different rows.

transistor A is broken into four smaller transistors with a width of ¼W. Another option is to split the transistor A in two smaller transistors of ½W and combine them as in Fig. 4c; finally, we can use two transistors of two fingers of ¼W width but placed in two different rows as in Fig. 4d. If the terminals of all the devices are correctly connected, the three implementations will have the same W_{eff} width.

4.3 Row Calculation and Placement Generation

For the placement implementation, the scripts used by the AMG are based on those used in [1]; however, a newly added function is the "Row calculation" one (see Fig. 3), which is used in the DB-mode, and it calculates the number of rows that are required for the layout implementation. This calculation is based on the number of fingers needed, and the number of transistors (in the case of an array of stacked devices), which are updated in each iteration; it also depends on the maximum layout width and height allowed (or layout area). This function is not used in the SL-mode since in this option the user directly defines the desired number of rows.

Once the information about the number of rows and devices has been updated, the rest of the scripts that form the

placement process are executed (see Fig. 3):

a) Validation of the number of transistors: here, the layout is validated for its proper implementation, which is based on the number of rows and the number of transistors (the number of rows should be divisible by the number of transistors). If it is not possible to draw the layout, an error is flagged, the topology is reported as invalid, and the rest of the scripts are executed. An implementation is considered invalid when the devices cannot be distributed uniformly among the calculated number of rows or when the number of fingers in which the transistor is divided does not allow the use of a specific topology.

b) Topology setting: this function uses the pattern definition templates to define the order in which the devices are laid out. The template is replicated, as needed, to cover the total number of transistors of the layout to be implemented. With this template, a list that indicates the order in which each device should be placed is created. This order depends on the module, the topology, the total number of transistors, and the number of rows (determined by the "Row calculation" function).

c) Placement drawing: this function creates individually each device following the order defined in the previous block. This function also names, rotates and flips[4] the devices. The Transistor Generator [1] algorithm creates every single transistor by drawing it on a specific location. The Placement Drawing Control sub-function dictates where to draw each device. The AMG draws transistor by transistor and row by row, following the order defined by the Topology Setting function.

4.4 Errors Report

Once all the above scripts are executed, the next step checks if an invalid layout is obtained, in which case, errors are reported (see Fig. 1). Errors can be found by the "Validation of number of transistors" function and by the "placement drawing" function; in both cases, a detailed errors report file is generated (see Fig. 3). In SL-mode, errors are displayed on a message window, while in the DB-mode are sent to a filed list of errors. If no error is found in the layout, the LAYOUT COMPLETION section is executed and the layout is included in a document with the list of valid layouts.

5. Layout Completion

This process is the same as that one presented in [1]. Here, all the metal routes and contacts are added, depending on the layers selected through the GUI.

5.1 Metal Routing Process

Pattern templates guide this metal routing. Each of the main structures and their topologies have their own pattern template and are based on predefined routing grids and the corresponding technology process design rules. In the DB-mode, it is recommended to include the base layers and the metals for interconnections. The tool is still limited to M2 for horizontal routing and M3 for vertical routing. In future work, if more complex modules or structures are included, and thus require the use of more (higher) metal layers, these can be easily implemented. Once the layout is completed, the module name with its corresponding characteristics are saved in a list of valid layouts (see Fig. 1). Two examples of layouts (differential pair) using different numbers of fingers and different finger's widths are shown in Fig. 5. In Fig. 5a) the differential pair's transistors are formed by two fingers, the finger's width is $2W_{min}$, and the layout is placed in one row. In Fig. 5b, the layout is placed in two rows, the transistors are formed by 24 fingers, and the finger's width is W_{min}, The layouts are fully connected (no opens) and DRC clean for spacing rules. Each row has two dummy devices at each side.

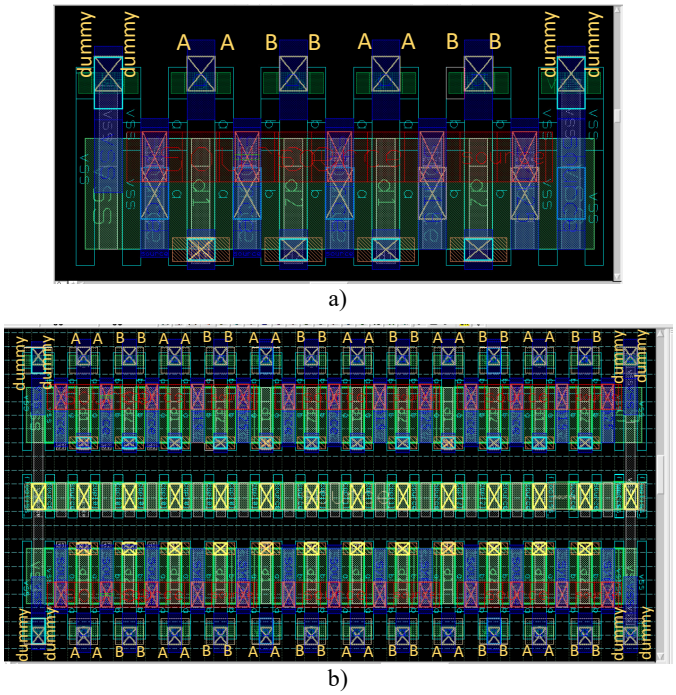

Fig. 5 Layout examples of a differential pair using different number of fingers rows and finger's widths: a) transistors are formed by two fingers, the finger's width is $2W_{min}$, and the layout is placed in one row; b) transistors are formed by 24 fingers, the finger's width is W_{min}, and the layout is placed in two rows.

5.2 Termination Criteria and Report Generation

Once the layout is completed, it is necessary to verify if the tool has reached the termination criteria (in the SL-mode this is not needed). If it is the case, the next step is the generation of a report (library) identifying the layouts created and their characteristics, which are saved in a document that summarizes this information. In SL-mode, this information is also presented in a message window.

If the termination criteria are not yet satisfied, the "Variables increment" block updates the values for creating a new layout, starting from the number of fingers of each device and then the transistors' sizes and in the case of an array of stacked devices, the number of devices.

6. Extra Flows

Once the termination criteria are satisfied, and the reports

[4] Rotation and flipping is automatically realized by the algorithm as needed to correctly complete the structure.

mentioned above are generated, the parasitics extraction algorithm and the LVS flow can be optionally executed over all the elements saved in the database following the list of valid layouts.

6.1 Parasitics Extraction and Area Calculation

The parasitics extraction algorithm mentioned in [1], which was available as a separate tool, has now been fully incorporated in the AMG. All the information related to the parasitic elements (restricted to parasitic resistance and capacitance in the current version) is obtained from the report files that the extraction algorithm generates. In the SL-mode, the user can directly read the complete report. However, when multiple layouts are generated, the manual analysis of all the reports can be time-consuming and error-prone. AMG filters out the data of these reports to obtain the information related to some specific parameter. For instance, the AMG can automatically obtain from these files the total input capacitance and the cross capacitance, as well as the sum of capacitances for particular nodes to the VSS (substrate) node. As part of the extraction flow (see Fig. 1), AMG calculates the required area for the implementation of each of the layouts contained in the generated library. Once the extraction flow is finished for all the layouts, a report is generated indicating the topology, the capacitance values, and the required area.

6.2 LVS Flow

In the DB-mode, this flow runs over all of the layouts created to detect any possible problem with the layout implementation. It only verifies that the layouts are open and short-circuit clean; since the netlists are based on the created layouts, this ensures they are LVS compliant. Once the LVS flow checked all the created layouts a file with the results is generated, indicating which layouts are clean and which are not; in this second case, the cause of the errors is also reported. For our current version, only LVS flow is included; the options for the users to run other flows can be added to the tool such as DRC or density checkers;

Once this flow is finished, and all the reports are generated, the AMG stops, and the user can review the results.

7. Test and Results

The AMG is now tested by generating one library for each of the two common analog modules mentioned before.

7.1 Database Creation

To create the database, the termination criteria are set up to enable the creation of the layouts using from one to three rows for each layout implementation. Another termination criterion is defined by using a layout width value such that the maximum number of devices in a row is thirty (including the dummy devices, which are two at each side of the array). The selection of these values has the intention to test the different capabilities of the proposed AMG, including its capacity to distribute the transistors uniformly among a different number of rows, addressing those cases where a given distribution is not possible.

Other considerations for the AMG test are the transistors finger's widths and the topologies for each module. Given the nanometric technologies considered, the dimensions used for the transistor are discrete values: the finger's widths used for the transistors' implementation are two, four, and six times W_{min} (the minimal feasible value for transistor's width for a given technology). Only one topology is used for each module: for the case of the differential pair, the AMG uses the interdigitated layout implementation; in consequence, the total number of fingers is an even number. For the case of the array of stacked devices, the topology used is the shared diffusion, and for this topology, the number of fingers of each transistor must be an odd number [1].

TABLE I
Summary Of Results For The Generated Libraries

Parameter Evaluated	Differential Pair	Stacked Array
Number of valid layouts	38	98
Number of invalid layouts	7	29
Percentage of valid layouts	84%	77%
Maximum capacitance to VSS	7.34 pF	9.35 pF
Maxim number of devices	90	77
Maximum number of rows	3	3
Maximum required area:	9.0521 μm^2	9.978 μm^2
Average implementation time per layout	98 seconds	113 seconds

For each database generated during these tests, a report of valid and invalid layouts (based on the number of transistors and rows) is automatically produced, as well as the corresponding summary report of other quantitative features. For all the valid layouts, extraction and LVS flows are executed (see Section VI).

Table I presents the results for the generation of both libraries, including the total number of valid and invalid implementations. It is seen that there are much more valid implementations for the library of stacked devices than for the library of differential pairs; this is due to the extra variable used in their implementation (the number of devices on the array). However, the proportion of invalid implementations is larger in the case of stacked devices since it is more difficult to distribute all the elements of the array between the different rows and accordingly to the selected topology.

It is also seen in Table I that the total capacitance to the VSS node is larger for the case of stacked devices since the complexity of the routing is higher (a deeper analysis is presented in the next subsection). Regarding the average implementation time per layout, we can notice that for both cases it is shorter than two minutes. This allows the generation of a complete library of approximately 100 elements, including parasitics, in less than two hours.

7.2 Parasitics Analysis

An analysis of different layout implementations of an array of stacked transistors is presented in [4], comparing the tradeoffs between different topologies and different layout parameters in term of their parasitic elements. The layout implementations used for that analysis are generated using the automatic synthesis tool in [1]. A similar procedure is

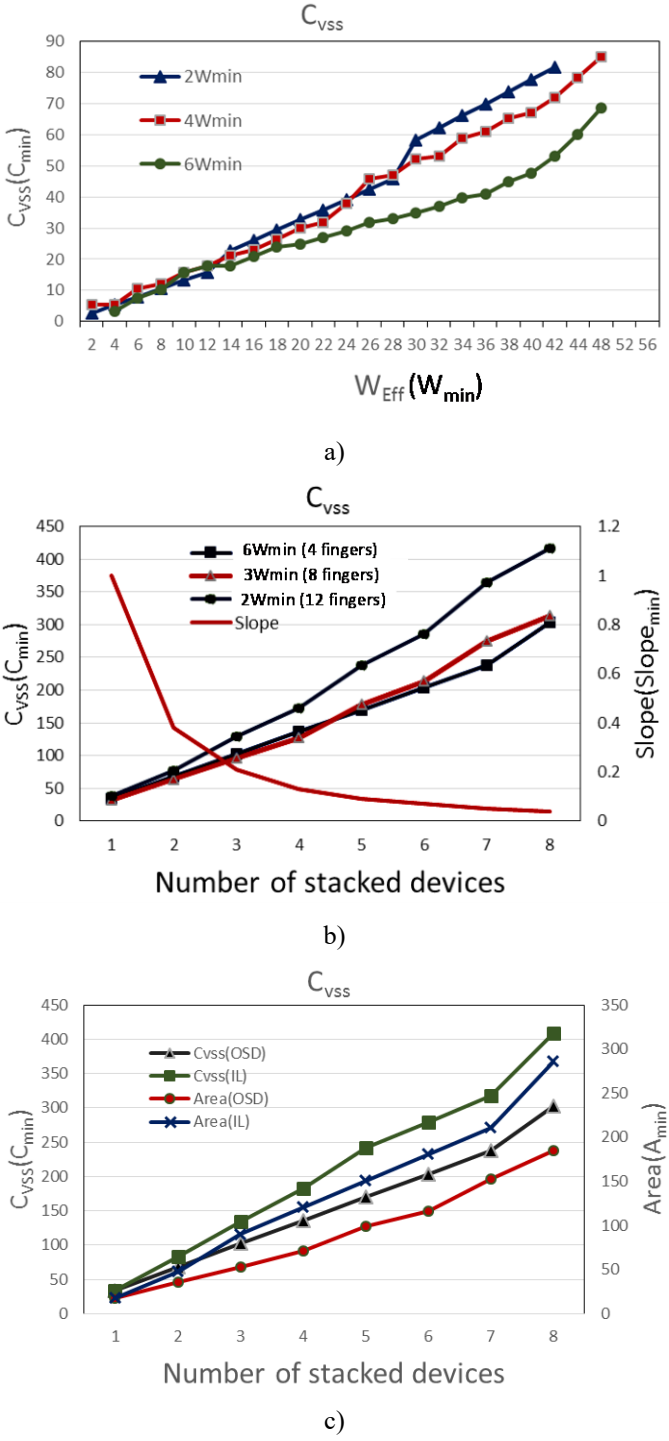

Fig. 6 Comparison of the C_{vss} capacitance value for: a) differential pair using different W_{eff} and varying the finger's width; b) array of stacked devices using a different number of transistors and varying the number of fingers; c) array of stacked devices using two different topologies: the interdigitated layout (IL) and the one shared diffusion (OSD).

performed here to show the benefits of using the new proposed AMG. The algorithm for parasitics extraction is now executed over all the valid layouts generated. The sum of the capacitances from all the nodes of each circuit to VSS node (C_{vss}) [4] is obtained, as well as the required areas for the implementation of each layout. These data are automatically organized in the parasitics report and can be used for a systematic comparison between all the different layouts, which illustrates the information that our CAD tool can provide to the designers.

To demonstrate the above mentioned capabilities, Fig. 6 shows three graphs that depict a comparison of C_{vss} values for both fundamental analog structures, considering different values for the number of fingers/devices, finger's widths, and the number of transistors in the arrays. In all these cases, the C_{vss} values are expressed in terms of C_{min} (the C_{vss} capacitance for a single transistor using minimal dimensions).

Figure 6a shows C_{vss} values for the differential pair module when different effective widths are used. For this example, and for illustration purposes, we parsed from the report file the cases for finger's widths equal to $2W_{min}$, $4W_{min}$, and $6W_{min}$. Naturally, as W_{eff} increases, C_{vss} increases as well. However, in Fig. 6a it is observed that for large W_{eff} (in this example larger than $14W_{min}$) the capacitance value tends to be smaller when the finger's width is larger (or when less fingers are used).

This indicates that for a large W_{eff} if we aim at reducing C_{vss}, it is better using wider transistor's values than increasing the total number of devices/fingers. Increasing the number of devices increases the total diffusion area used for the devices, which increases the capacitance between all of them; also as we use more fingers, more rows are required for their placement, which increases the number of interconnections and their length, as well as the capacitance associated to them.

Figure 6b shows C_{vss} values when a different number of stacked devices and finger's widths are used on an array, but keeping $W_{eff} = 24W_{min}$ for all cases. The cases parsed from the reports are for a finger's width equal to $2W_{min}$, $3W_{min}$, and $6W_{min}$ (which correspond to 12, 8 and 3 fingers respectively). As in the previous case, when larger finger's widths are used, the C_{vss} values are smaller than when more fingers are used. Additionally, when more devices are added to the layout, more rows may be required for their placement, and more interconnections are needed, which increases the number of metallic wires required. In consequence, the required area for layout implementation is larger when transistors are divided into fingers than when large finger's widths are used, as it was confirmed in [4].

In this example, reducing the number of fingers, help to reduce the generation of parasitic elements, something designer should consider for the implementation of their circuits. Other aspects designers should also consider are, for example, the fact that when a transistor is divided its switching speed can increase; the use of multiple fingers allows a better matching and help to reduce current density in the gate.

Fig. 6b also shows the slope values, which represent the output resistance of the transistor. It is seen that, as the number of stacked devices increases, the improvement on the output resistance is less significant, but the total capacitance increases almost linearly (as it was also found in [4]). In summary, Fig. 5b illustrates the combined effects that different layout implementations have over total parasitic values and output impedance.

Finally, Fig. 6c shows C_{vss} values and the required layout

area varying the number of stacked devices on the array module and considering two topologies: the Interdigitated Layout (IL) and the One Shared Diffusion (OSD) [1]. For both topologies, the finger's width is $2W_{min,}$ and the number of fingers is 3. Naturally, the topology determines the length and location of the metals used for interconnection of the devices, and the different metals' lengths affect the values of the parasitic elements, as it is confirmed in Fig. 5c. Since the OSD topology requires less and shorter metal wires, the values of its parasitic elements are smaller than for the IL topology (see Fig. 6c). Similarly, since the OSD topology employs fewer wires than the IL topology, it requires smaller layout implementation areas (see Fig. 6c).

The above graphical examples illustrate some of the features and capabilities of the proposed AMG. These examples also show the main two advantages of the proposed AMG tool, as compared with other approaches. Firstly, the automatic generation of numerical data related to different layout properties, such as parasitic components and area, useful for further statistical analysis. Secondly, the capability of generating analog layouts, comparable in quality to those manually done by layout experts, but in a much faster manner (a few minutes instead of several hours).

8. Conclusions

In this work, we described the general features of an analog modules generator (AMG) tool intended to accelerate the layout implementation of fundamental analog circuits and enable the generation of layout databases.

The tests presented show the efficiency and utility of the proposed AMG for generating multiple versions of common analog modules integrated into libraries. The results of the parasitics extraction process, executed over all the layout libraries created, were also presented, illustrating the information that the reports generated by the AMG can provide to the designer. These experimental results show that this new layout tool is capable of producing many layout versions of the same structure. In contrast to other approaches, the proposed AMG is capable of identifying problems in the implementation of the layouts, and generating reports with information related to their area and parasitics components in a very short amount of time. Users can process this information straightforwardly, and develop a statistical analysis to make performance comparisons and select the alternative that best meets their requirements. Results show that all these features will help to reduce the design effort for analog circuits.

This work offers the possibility of some future research opportunities. In its current version, our tool creates all possible valid layouts in a database for a specific constraint: the available area. However, the tool can be enhanced by incorporating other constraints, such as the maximum C_{vss} parasitic capacitance, or the maximum number of metal layers, etc. Another relevant future work consists of using numerical optimization techniques to find the best layout implementation according to some user-defined design specifications.

Acknowledgment

Authors thank Víctor H. Martínez-Sánchez, from Intel Corp., Zapopan, Mexico, who greatly assisted this research.

References

[1] I. Lomelí-Illescas, S. A. Solis-Bustos, V. H. Martínez-Sánchez, and J. E. Rayas-Sánchez, "Synthesis tool for automatic layout generation of analog structures," in *IEEE ANDESCON Proc*., Arequipa, Peru, Oct. 2016, pp. 1-4.

[2] Y. Ender and D. Günhan, "Analog layout generator for CMOS circuits," *IEEE Trans. Computer-Aided Design Integrated Circuits Systems*, vol. 28, no. 1, pp. 32-45, Jan 2009.

[3] C. Pacha et al., "Circuit design issues in multi-gate FET CMOS technologies," in *IEEE Int. Solid State Circuits Conf. - Digest of Technical Papers*, San Francisco, CA, 2006, pp. 1656-1665.

[4] I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, "Analysis of the implications of stacked devices in nano-scale technologies for analog applications," in *IEEE Latin American Test Symp. (LATS-2017)*, Bogota, Colombia, Mar. 2017, pp. 1-4.

[5] D. H. Saari and D. G. Nairn, "Analog integrated circuit design using fixed-length devices," in *IEEE Int. Symp. on Circuits and Systems (ISCAS-2016)*, Montreal, QC, May 2016, pp. 1798-1801.

[6] F. Balasa and K. Lampaert, "Module placement for analog layout using the sequence-pair representation," in *Proc. Design Automation Conf.*, New Orleans, LA, Jun. 1999, pp. 274-279.

[7] S. W. Mehranfar, "STAT: a schematic to artwork translator for custom analog cells," in *IEEE Proc. of the Custom Integrated Circuits Conf.*, Boston, MA, May 1990, pp. 30.2/1-30.2/4.

[8] J. Canaris, "Schematic driven layout for the custom VLSI design environment," in *Proc. First Great Lakes Symp. VLSI, Kalamazoo*, MI, Mar. 1991, pp. 302-306.

[9] H. Graeb, F. Balasa, R. Castro-Lopez, Y. Chang, F. Fernandez, Po-hung Lin, and M. Strasser, "Analog layout synthesis -recent advances in topological approaches," in *Conf. Design, Automation and Test in Europe*, Belgium, 2009.

[10] Q. Ma, L. Xiao, Y. C. Tam, and E. F. Y. Young, "Simultaneous handling of symmetry, common centroid, and general placement constraints," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 85-95, Jan. 2011.

[11] P. H. Wu et al., "Performance-driven analog placement considering monotonic current paths," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD-2012*), San Jose, CA, Nov. 2012, pp. 613-619.

[12] N. Lourenco, M. Vianello, J. Guilherme, and N. Horta, "LAYGEN—automatic layout generation of analog ICs from hierarchical template descriptions," *Ph.D. Research in Microelectronics and Electronics*, Otranto, Italy, Sep. 2006, pp. 213-216.

[13] R. Martins, N. Lourenço, and N. Horta, "LAYGEN II—automatic layout generation of analog integrated circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 11, pp. 1641-1654, Nov. 2013.

[14] A. Mohamed, M. Dessouky, and S. M. Saif, "Analog layout placement retargeting using satisfiability modulo theories," in *Int. Conf. on Synthesis, Modeling, Analysis and Simulation Methods and Applic. to Circuit Design (SMACD),* Giardini Naxos, Italy, Jun. 2017, pp. 1-4.

[15] P. H. Wu, M. P. H. Lin, T. C. Chen, C. F. Yeh, X. Li and T. Y. Ho, "A novel analog physical synthesis methodology integrating existent design expertise," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 199-212, Feb. 2015.

[16] D. Marolt, J. Scheible, G. Jerke, and V. Marolt, "Analog layout automation via self-organization: enhancing the novel SWARM approach," in *IEEE 7th Latin American Symp. on Circuits & Systems (LASCAS)*, Florianopolis, Mar. 2016, pp. 55-58.

[17] M. Strasser, M. Eick, H. Grab, U. Schlichtmann, and F. M. Johannes, "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions," in *IEEE/ACM Int. Conf. on Computer-Aided Design*, San Jose, CA, Nov. 2008, pp. 306-313.

[18] L. Xiao and E. F. Y. Young, "Analog placement with common centroid and 1-D symmetry constraints," in *Asia and South Pacific Design Automation Conf.,* Yokohama, Japan, Jan. 2009, pp. 353-360.