INSTITUTO TECNOLÓGICO
Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

DOCTORADO EN CIENCIAS DE LA INGENIERÍA

# DESARROLLO DE UNA HERAMIENTA DE CAD PARA LA GENERACION AUTOMATICA DE BIBLIOTECAS DE *LAYOUT* DE CIRCUITOS ANALOGICOS BASICOS

Tesis que para obtener el grado de

DOCTOR EN CIENCIAS DE LA INGENIERÍA

presenta: Ismael Lomelí Illescas

Director de tesis: Dr. Sergio Alfredo Solís Bustos

Co-director de tesis: Dr. José Ernesto Rayas Sánchez

Tlaquepaque, Jalisco. Diciembre de 2018

DOCTOR EN CIENCIAS DE LA INGENIERÍA (2018) ITESO, Tlaquepaque, Jal., México

| | |
|---|---|
| **TÍTULO:** | **Desarrollo de una herramienta de CAD para la generación automática de bibliotecas de *layout* de circuitos analógicos básicos** |
| **AUTOR:** | Ismael Lomelí Illescas<br>Ingeniero en Comunicaciones y Electrónica (Universidad de Guadalajara, México)<br>Maestro en Diseño Electrónico (ITESO, México) |
| **DIRECTOR DE TESIS:** | Sergio Alfredo Solís Bustos<br>Doctorado en Ciencias (INAOE, México)<br>Maestría en Ciencias (INAOE, México)<br>Licenciatura en Electrónica (BUAP, México)<br>Gerente de Diseño Fisco (*Intel Corporation* Campus Zapopan) |
| **NÚMERO DE PÁGINAS:** | xxv, 112 |

ITESO – The Jesuit University of Guadalajara

Department of Electronics, Systems, and Informatics

DOCTORAL PROGRAM IN ENGINEERING SCIENCES

# DEVELOPMENT OF A CAD TOOL FOR THE AUTOMATIC GENERATION OF COMMON ANALOG LAYOUT STRUCTURES AND LIBRARIES

Thesis to obtain the degree of

DOCTOR IN ENGINEERING SCIENCES

Presents: Ismael Lomeli-Illescas

Thesis Director: Dr. Sergio Alfredo Solís-Bustos

Thesis Co-director: Dr. José Ernesto Rayas-Sánchez

Tlaquepaque, Jalisco, Mexico
December 2018

PhD in Engineering Sciences (2018) ITESO, Tlaquepaque, Jal., Mexico

**TITLE:** **Development of a CAD Tool for the Automatic Generation of Common Analog Layout Structures and Libraries**

**AUTHOR:** Ismael Lomeli-Illescas
Bachelor’s degree in communications and electronics engineering (*Universidad de Guadalajara*, Mexico)
Master’s degree in electronics design (ITESO, Mexico)

**THESIS DIRECTOR:** Sergio Alfredo Solís-Bustos
Physical Design Manager (Intel Corporation Campus Zapopan)
Bachelor’s degree in electronics (BUAP, Mexico)
Master’s degree in electronics (INAOE, Mexico)
Ph.D. degree in electronics (INAOE, Mexico)

**NUMBER OF PAGES:** xxv, 112

This work is dedicated to my father, Ismael, and my mother, Araceli,
for their understanding, patience, and the continuous loving
support they gave me during all the time that this
important project in my life lasted.

# Resumen

La implementación física o “layout” de circuitos analógicos es una parte crítica en el proceso de diseño de circuitos integrados (CI). El *“*layout*”* de circuitos analógicos en tecnologías de fabricación nanométricas típicamente utiliza transistores con características digitales, lo cual impone uno de los principales desafíos en el diseño de CI debido a las limitaciones en las dimensiones del transistor (restringidas a la utilización de múltiplos de los valores mínimos de longitud y ancho de dichas tecnologías). Además, los circuitos analógicos deben cumplir con especificaciones muy estrictas de diseño, tales como: operación a alta frecuencia, bajo nivel de ruido y alta precisión, las que a su vez dependen de la implementación física. De aquí que la creación apropiada del “layout” es críticamente importante en el desarrollo de CI analógicos. Por otro lado, las herramientas de diseño asistido por computadora (CAD, por sus siglas en inglés) para el diseño físico de CI analógicos están aún lejos de alcanzar un estado de madurez, en contraste con aquellas para el diseño físico de CI digitales. Esto debido a que el diseño analógico es menos sistemático y más heurístico que el diseño digital, por lo que requieren conocimientos y habilidades de diseño más especializados. Adicionalmente, los CI analógicos son más sensibles a perturbaciones, elementos parásitos, interferencia electromagnética, ruido de sustrato y otras fuentes de ruido. Además, existe una amplia diversidad de esquemas para la implementación de cada módulo analógico. Es por ello que la implementación óptima de los diseños analógicos puede requerir muchas iteraciones con la intervención del diseñador experto, e incluso de varios ciclos completos de rediseño, generando ciclos de desarrollo excesivamente largos y costosos. En esta disertación doctoral se presenta una novedosa herramienta de CAD que permite la generación automática de diferentes versiones de “layout” para estructuras analógicas específicas. La herramienta propuesta permite la creación eficiente de múltiples topologías de “layout” para la posterior generación de bibliotecas de dos de las estructuras analógicas más fundamentales: el par diferencial y los dispositivos apilados. Los diseñadores pueden utilizar esta base de datos para el análisis, caracterización y optimización de sus diseños. En la tesis doctoral se presentan análisis de las estructuras mencionadas, así como pruebas que ilustran la funcionalidad y capacidades de la herramienta de CAD propuesta para la creación de dichas bibliotecas de “layout” en un periodo muy corto de tiempo, ayudando a los diseñadores a reducir el ciclo de diseño del circuito.

# Summary

The layout implementation of analog circuits has become a critical part of the design process of integrated circuits (IC). The physical construction of this kind of circuits using transistors with digital characteristics as the only devices available in many nanoscale fabrication technologies is one of the main challenges given the limitations in the transistor's dimensions, which are constrained to a multiple of the technology's minimum width and length. In addition to size limitations, analog circuits have to fulfill rigorous design specifications, such as high frequency performance, low noise, and high accuracy, which are strongly dependent on their physical implementation; hence, the optimal layout implementation of the analog circuit becomes of paramount importance in the design process of an integrated circuit. On the other hand, computer-aided design (CAD) tools for analog IC physical design are far from being mature, in contrast to those used for digital IC physical design. Some of the reasons for this are that analog design, in general, is less systematic and more heuristic in nature than digital design. Additionally, analog design often requires specialized knowledge, design skills, and years of experience; analog circuits are more sensitive to parasitic disturbances, EM crosstalk, substrate noise, supply noise, etc.; besides, the variety of schematics and diversity of devices and shapes are much more significant. For all these reasons, the optimal implementation of analog layouts requires several iterations and sometimes rework of the layout, resulting in a very long and expensive developing cycle. In this doctoral dissertation, a novel CAD tool is presented that enables the creation of different layout versions for selected analog structures. The main purpose of the proposed CAD tool is the automatic generation of multiple layout topologies for the subsequent generation of layout libraries or database of two of the most fundamental analog structures: the differential pair and the array of stacked devices. Circuit designers can use this database for the analysis, characterization, optimization, and suitable implementation of their designs. The present doctoral thesis describes several tests and studies for the structures mentioned above, illustrating the functionality and capabilities of the proposed CAD tool for the creation of multiple layout topologies in a very short time, helping designers to reduce the circuit's design cycle.

# Acknowledgements

The author wishes to express his sincere appreciation to Dr. Sergio Alfredo Solis-Bustos, from Intel Corp. campus Zapopan, for his encouragement, expert guidance and keen supervision as doctoral thesis director throughout the course of this work. The author offers his gratitude to Dr. José Ernesto Rayas-Sánchez, Professor of the Department of Electronics, Systems, and Informatics at ITESO, for his reviews and suggestions along the development of the internal research reports, conference and journals papers, as well as for the development of this document.

Author also thanks Dr. Esteban Martínez-Guerrero, Dr. Manuel Salim-Maza, and Dr. Víctor Avendaño-Fernandez, members of his Ph.D. Thesis Committee, for their interest, assessment, and suggestions.

Special thanks are due to Victor Hugo Martinez, formerly from Intel Corp. Zapopan and now at Oracle Guadalajara, for his fruitful cooperation in the development of multiple scripts that enabled the completion of the CAD tool presented in this work.

The author gratefully acknowledge the financial support through a scholarship granted by the *Consejo Nacional de Ciencia y Tecnología* (CONACYT, Mexican Government), as well as, the time flexibility, economic support, and computing tools provided by Intel Corp. campus Zapopan.

Finally, special thanks are due to my family for their understanding, patience, and continuous loving support.

# Contenido

# Contents

# List of Figures

# List of Tables

# Introduction

In current nanoscale process technologies (beyond 32 nm), one of the main challenges in the area of the analog circuit design is the implementation of high performance circuits using devices intended for digital applications. The reduction in the power supply voltage, as well as the reduction in transistor's size, impose stringent constraints in analog design, leading to long design time and effort [Ender-09]. In this context, analog layout design is strongly constrained by the transistor's dimensions, connections, and topologies used, and thus the optimal layout implementation becomes critical to achieving the desired high performance of the circuits.

With the usage of nanoscale CMOS technologies, analog designers are facing many new challenges in the layout implementation of the circuits. One of those layout challenges is the severe degradation in device matching characteristics as a consequence of the limitations of actual lithographic techniques: since current device sizes are less than one-quarter of the wavelength of the deep ultraviolet illumination (UV, $\lambda$= 193 nm), then more complex interference techniques are necessary for the actual construction of layouts, generating new random variances in layout dimensions. Another example is the high leakage currents resulting from low thresholds and thin gate oxide tunneling currents. In consequence, to accomplish electromigration limits, we require robust metal grids to connect shared-drain analog devices [Lewyn-09]. With all these problems in current layout technologies, the layout implementation become critical, making necessary to have robust layouts, capable of diminishing these effects.

In Chapter 1 the author analyzes the challenges in the implementation of analog circuits in actual nanoscale processes, mostly due to the restrictions in the sizing of the devices: the discretization on the transistor width, and fixed length values. In this same chapter a study of the layout implementation of two well-known circuits: the differential pair and the array of stacked devices is presented.

Chapter 2 presents an in-depth analysis of the implications of the performance of analog circuits due to the use of stacked devices in current nanoscale technologies. To evaluate the use of stacked devices, the characteristic curves of transistors implemented with a different amount of transistors in stack are obtained and compared to those of a single device. Even the analysis of the use of stacked devices is not the main purpose of this work; however, it is important to explain the importance of the use of these structures in the implementation of analog circuits. A similar analysis can be developed for the differential pair structure, however being a well-known structure and for ease of simplicity, this analysis is not presented in this thesis.

The analyses presented in Chapter 1 and Chapter 2 show that even for these basic structures there are several options for their layout implementation, each of them with numerous trade-offs.

In this context, to implement and evaluate different layout placements (or topologies) 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. However, a fully automated analog design is far from being mature. Analog IC design is still performed mostly manually. Full-custom analog designs lead to long design-test cycles, increasing the time to market and the overall cost [Graeb-09].

Several heuristic placement techniques have been implemented. Among the most popular is the constructive approach, which consists of gradually evolving the placement solution by selecting one module at a time and positioning it in the "best" available location [Graeb-11]. Several CAD systems for analog placement based on constructive methods have been reported [Canaris-91], [Balasa -99]. For instance, a schematic-driven approach employing connectivity and relative positioning in the input schematic is proposed in [Mehranfar-90]. Constructive methods are fast and scale well with the problem size; however, their main drawback is the dependence on the order of selection of the devices [Graeb-11].

Another common technique is based on constraints definition [Graeb-09], used either to define the location and placement of the different devices [Q. Ma-11] or to define the routing paths and, in consequence, the location of the different components [H.-Wu-12]. In general, most of these approaches require intensive user-tool interaction, as well as significant user's experience in analog layout design.

Other useful 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 [Lourenco-06]. The main disadvantage of this technique is that the generators require considerable coding effort for each new topology [Graeb-11]. Some variations of this approach have been proposed: a CAD tool intended to imitate the constructive layout style of "manual designers” by generating simple geometries and packing them together into a complex layout is presented in [Lihong-04]; while in [Yilmza-09] the designer directly interacts with the tool at different phases. A performance-constrained parasitic-aware retargeting tool can be found in [Zhang-10]. In [Unutulmaz-10], linear programming is used to generate analog layouts from simple declarative statements, similarly to digital implementations.

All of these implementations require a high level of user intervention for the placement process, and they need from the user considerable expertise in analog layout.

Another one employs the layout retargeting technique that consists of generating a new layout from an existing one. This approach defines the relative position, and the interconnect paths of the devices through the use of templates to guide the generation of the new layout typically used in the design migration from one technology process to a new one [Martins-13]. Examples of the layout retargeting technique are presented in [Mohamed-17], where foundry Parametric Cells (pCells) are used; in [H.-Wu-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.

In Chapter 3, the proposed analog layout synthesis tool is presented. The implementation is focused on two of the most commonly used analog building blocks: differential pairs and arrays of stacked devices. Starting from the complete circuit netlist and the names of the selected transistors, our tool verifies that the provided transistors form a valid building block and creates the corresponding layout. The user can define different layout parameters, such as transistor dimensions, number of rows, and number of dummy devices. The layout view can be generated with different levels of detail, including placement of the devices only, basic metals, metallization for complete interconnects, or design for manufacturing (DFM) compliance by adding dummy fills, guard rings. The proposed CAD tool aims at making the full design process more efficient with modern IC fabrication technologies, by providing the circuit designers with quick custom layout views that can be employed to simulate and optimize their designs.

In this same Chapter 3, several tests are presented to show the efficiency and utility of the proposed CAD tool for the generation of the layout of these analog structures, using different configurations and parameters. Also to illustrate how the designer can use the tool, for the analysis and comparison of the different layout implementations in terms of their parasitic components an extraction process is performed over the generated layouts.

The different test presented in Chapter 3 illustrates the advantages of the generation of parametric layouts; however, each of these implementations was generated individually (the tool was manually configured for each new layout) and the extraction process was executed individually over each generated layout. Based on the proposed CAD described in Chapter 3, an enhanced CAD tool that enables the creation of a layout database (library) of the aforementioned analog circuit is presented in Chapter4. The primary purpose of the tool is the automatic creation of multiple layout versions of two common analog structures: the differential pair and arrays of stacked devices, for the subsequent generation of a layout library. The proposed CAD tool automatically generates multiple layout versions of these structures. The tool validates all the possible implementations, based on the number of devices and rows; when the layout is a valid one (meaning that it is feasible to implement it with the user input number of rows and devices); it is saved in a database, with its corresponding characteristics. Optionally, an extraction algorithm may be executed over all the elements saved in the database. Finally, the correct implementation of the different layout versions is validated by using a layout versus schematic verification flow. The CAD tool is tested by generating one library for each of the two common analog structures mentioned before. A list of valid and invalid layouts is created as well as the summary of results.

In the general conclusions, the most relevant contributions of the proposed CAD tool are summarized. Additionally, some ideas for future research work are presented. Finally, Appendix A shows the reference list of the eleven internal research reports that the author wrote during his doctoral studies, and Appendix B shows the list of conference and journal papers also published.

# 1. IC Layout Limitations in Nanoscale Fabrication Technologies

The limitations to implement the layout of analog integrated circuits (ICs) in current nanoscale technologies are described in this chapter. Special emphasis is placed on the constraints to size and physically implement transistor’s width and length, which are limited to discretized values as multiple of one minimal or reference size, determined by the fabrication technology. Additionally, some of the most important considerations for the layout implementation of reliable analog circuits, which include device matching, metal routing, device interconnection, and design rules are also discussed. Finally, an analysis that considers different options for the layout implementation of two widely used topologies in analog ICs, namely, the differential pair and arrays of stacked transistors is presented. The analysis is intended to show the opportunities that analog layout automation offers for improvement to give circuit designers the capability to test several layout options in less time making the layout design process more efficient and then contribute to the reduction of the IC design cycle and total cost.

In recent years, thanks to the evolution in IC fabrication technologies towards nanoscale CMOS technologies (90, 65, 45nm and beyond), the integration of complex systems, such as network interfaces, wireless designs, telecommunications, and multimedia systems in one single chip, named Systems on Chip (SoC) have been possible [Casier-11]. These integrated systems are increasingly mixed-signal designs, embedding high-performance analog or mixed-signal blocks and possibly sensitive RF frontends together with complex digital circuitry (multiple processors, some logic blocks, and several large memory blocks) on the same chip.

However, the use of CMOS (Complementary Metal Oxide Semiconductor) nanoscale technologies brings significant challenges for mixed signal design that were not encountered before that impact the physical design implementation (layout) of the systems. These challenges include:

a) The increasing variability of technology parameters, causing mismatch and yield problems.

b) Aggravating degradation mechanisms (e.g., NBTI-Negative Bias Temperature Instability-Hot Carriers) and increasing reliability constraints such as EMC (Electromagnetic Compatibility) and EMI (Electromagnetic Interference) regulations [Casier-11].

In addition to the above challenges, managing the ever-increasing design complexity in tightening time-to-market constraints makes the physical implementation of mixed signal designs and its verification more difficult, highly time-consuming and expensive. In order to enhance designer’s efficiency while reducing time and cost different EDA (Electronic Design Automation) methodologies and tools are available for the IC design industry.

While tremendous progress has been made in the digital layout field in recent years, the special characteristics of analog layout design make the development of flexible analog IP (Intellectual Property) blocks a highly difficult task [Saravanan-11].

Some of the main challenges and considerations associated with analog layout design, such as reliability and leakage current problems, were addressed in [Lomelí-Illescas-13]. In this reference, some opportunities for the implementation and improvement of analog layout automation tools are described and they are the source of inspiration for this research work. We base our research on the correct understanding of the constraints and requirements for the implementation and optimization of analog layout through the development of an analog layout automation tool that enables their fast generation and analysis.

In the following section we describe in detail the main limitations found in current nanoscale design technologies.

## 1.1. Limitations in Nanoscale Technologies

In many old fabrication technologies (micrometers range), the minimal transistor's width was limited by the resolution of the lithographic process, and almost there were no limitations for the maximal transistor's width, except for some practical considerations. In other few cases, the width needed to be a multiple of the device's length. However, in most of the current nanoscale technologies, there is a minimum dimension for the width of the device, which is known as the transistor's pitch (see Fig. 1.2). Larger transistor dimensions need to be a multiple of this pitch. Thus, we can say that in current technologies there is now a discretization of the devices' dimensions. Historically, the pitch scales down around by a factor of 0.7 every two years. This means that the density increases as the pitch decreases by square law. This scaling is sustained by Moore's law, as shown in Fig. 1.2.

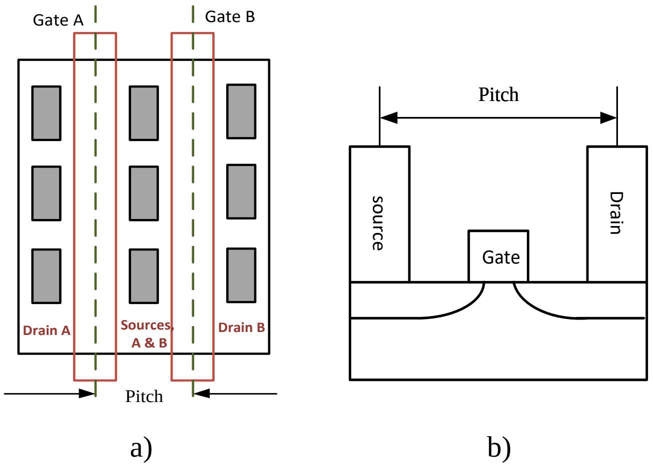


Fig. 1.1 Illustration of technology pitch: a) transistor top view or layout view, b) transversal cut of the transistor.

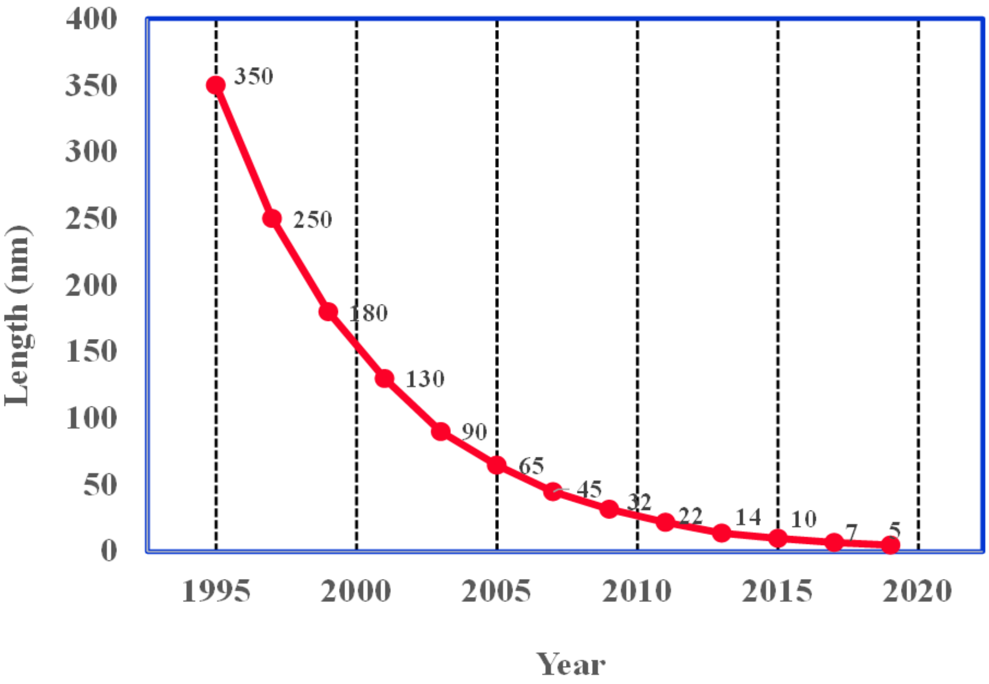


Fig. 1.2 Scaling of technology pitch defined by Moore’s law.

We also need to consider that transistor's width is not related to its length and even the minimum size is not optimal for the insertion of multiple contacts or vias, which can cause reliability

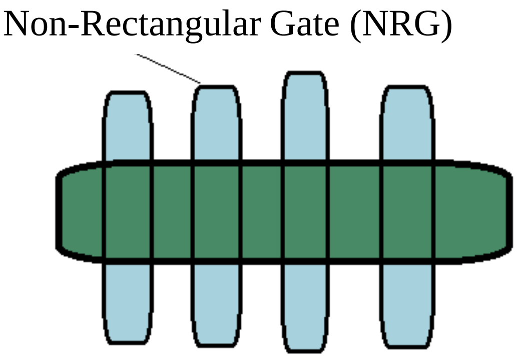


Fig. 1.3 Non-Rectangular Gate (NRG) effect. Example of distortion of a device gate shape.

problems on DC signals or Self Heating[1] (SH) problems on analog or AC signals.

Given all these restrictions in the sizing of the devices, the options for the implementation of analog structures are highly limited, similarly to the implementation of digital circuits, but with the requirements and constraints of the analog ones. In recent technologies, most of the non-desired effects are not exclusive for analog or digital layout implementations, but they are observed in both kinds of circuits [Maricau-13].

The transistor's width has impact on different design and physical parameters, for example, the threshold voltage ($V_t$), transistor's transconductance ($g_m$), and saturation current ($I_{dsat}$) matching. These parameters depend on the precision of many process-driven variables, such as bulk doping concentration, $V_t$-shifting implant dose and range, carrier mobility, gate-oxide thickness, and the device W/L dimensional accuracy. However for IC technologies starting at the 100nm node, additional local effects have become significant factors influencing transistor's matching. These effects include $V_t$ shifts resulting from the proximity of the gate to the N-well edge and $V_t$ and mobility shifts resulting from the distance to the local trench isolation, among others. At nanoscale technologies, the matching of the devices becomes critical due to second-order effects related that arise from the reduction of the transistor's dimensions [L. Lewyn-09].

Finally, regular layouts that follow restrictive design rules are essential to have robust CMOS designs to alleviate many manufacturing induced effects, such as the effect of a Non-Rectangular Gate (NRG) due to sub-wavelength lithograph (see Fig. 1.3). NRG dramatically increases the leakage current by more than 15X compared to that of an ideal physical layout; to

[1] Self-Heating (Joule Heating) is the rise in temperature of a line as alternating current (AC) flows through it.

mitigate such a penalty, some techniques have been proposed to guarantee regular layout through restrictive design rule parameters [R. Subramaniam-12].

## 1.2. Analog Layout Challenges

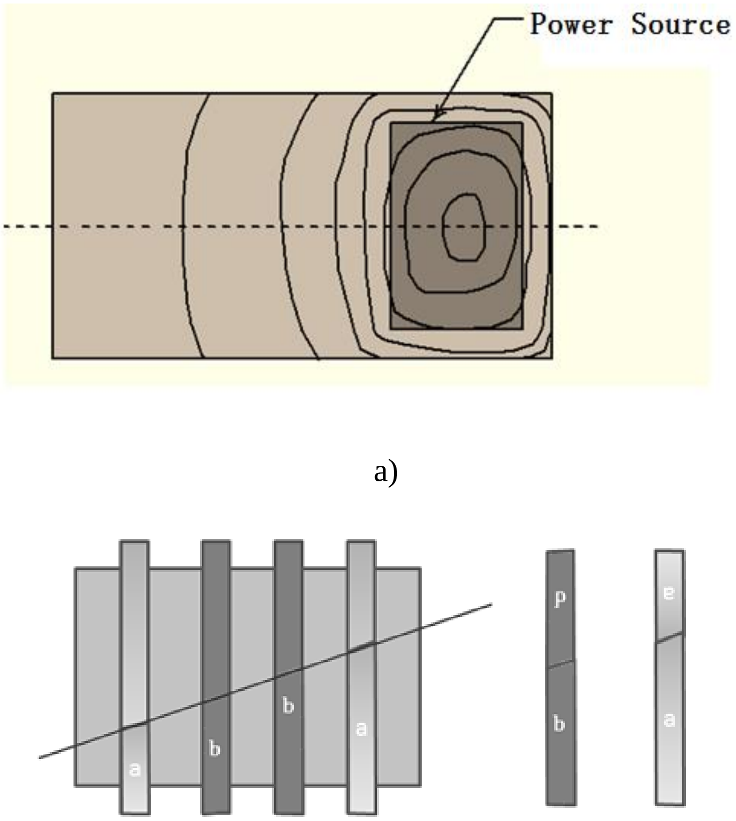
From previous section one of the main challenges in the area of the analog circuit design is the implementation of this kind of circuits using discretized devices; these devices are typically optimized for digital applications. Due to these limitations, an optimal layout implementation is essential to achieve the proper performance of the circuits.

For an optimal layout implementation in a given application, the primary factor that needs to be considered is the matching between devices. However, there are other factors and/or parameters that the designers should address and analyze their contribution on the layout implementation; some of these parameters are the metal routing and its connections (metal layer, contacts or vias, etc.). Similarly, Design Rules (DR) should also be considered; usually, these are not related to a fixed value, but to a range of them, so the election of an optimal value can contribute to diminishing the effect of the non-desired phenomena. These factors are becoming more important in current layout CMOS technologies due to the limitations brought by the scaling on the transistor's minimum size.

## 1.3. Matching

Historically, having proper matching between critical devices is one of the most critical layout design techniques to guarantee the correct functionality of circuits. In analog circuits, matching off voltages and currents is crucial, for example, an 8-bit Video Digital to Analog Converter (DAC) design requires that the current sources match to within +/- 1%; a differential transistor pair circuit typically requires a $V_t$ mismatch of less than 0.5 mV (≈0.1%).

Device matching in analog and mixed-signal integrated circuits must be carefully considered for high performance and yield. In a typical CMOS process, the absolute accuracy of components such as transistors, capacitors, and resistors varies by as much as 20%, while the parametric ratios may match to within ~0.1%. For this reason, analog circuit implementations typically rely on component matching rather than absolute accuracy. In addition to choosing



a)

b)

Fig. 1.4 Variations of temperature caused by a power source: a) illustration of the thermal gradient, b) matching technique of interdigitating, commonly used to mitigate this effect.

appropriate transistor dimensions for proper matching, the layout must also be carefully constructed [Ravindranath-98].

However, current technologies have made more difficult to achieve a proper matching: the devices and wires, are closer and the surrounding elements have a more significant effect on them.

Six factors affect transistor matching:

a) Coincidence: the centroid ("center of gravity") of the matched devices should coincide to minimize the impact of the variation on the different gradients. The gradients (process and temperature) have become critical due to the reduction of dimensions. The illustration of the gradient and an essential technique to mitigate it are shown in Fig. 1.4.

b) Symmetry: the structure should be symmetric around both the X and Y-axes. It also reduces the impact of gradient variations and enables symmetric routing.

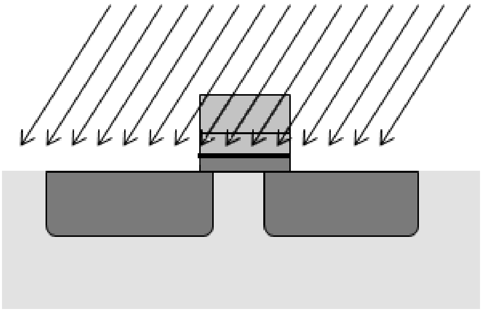


Fig. 1.5 Illustrating the Chirality effect.

c) Dispersion: the legs of each device should be distributed throughout the array as uniformly as possible to minimize the impact of local variation.

d) Compactness: the structure should be as compact as possible (ideally a square) to reduce mismatch due to separation.

e) Proximity (edge effect): each row and column should have an equal number of legs of each device to minimize mismatch from external signal routing or thermal gradient (edges of the array need to be balanced).

f) Chirality: each device should possess same chirality to minimize the impact of implant angle and orientation-dependent process variation. Chirality is a measure of left-handedness vs. right-handedness. In other words, each device should have an equal number of legs with source on the left. One form of ideal chirality can be achieved if all legs are arrayed in pairs, as illustrated in Fig. 1.5.

As it can be noticed, the different kinds of gradients affect the matching of the devices. Thus, we may use compact layouts to prevent it, however this could cause thermal effects, as the devices are closer to each other.

Techniques applied heuristically, like dummy devices added on arrays of transistors certainly improve matching but need to be quantified to be implemented correctly. The use of dummy devices is detailed next.

### 1.3.1 Use of Dummy Devices (Dummification)

Many of the characteristics of a layout depend on the surrounding environment. During the

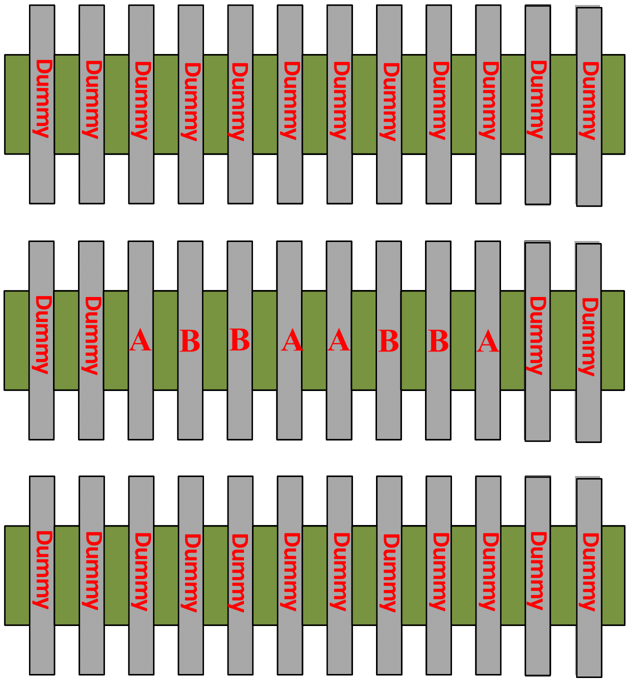


Fig. 1.6 Ideal dummification. Dummies placed around matched devices to get a similar surrounding environment.

manufacturing process, due to the mechanical polish rates, gas etches rates, material deposit thickness, etc., the device length and width, the implant dose, and the terminal resistance can be impacted; to optimize device matching, each device's environment should be as identical as possible, that is why it is necessary the inclusion of dummy devices at the sides of the transistor arrays, as shown in Fig. 1.6. The use of these dummy transistors is, in consequence, very common but not only for matching purposes of two particular devices but to match the size of a specific section, with another one, inside the complete analog circuit. Furthermore, it is common to add multiple dummy devices to prevent future changes in the design like enhancing current driving capability of the circuit.

The analysis of the different layout topologies could help to optimize the number of dummy transistors and to optimize the area of the layout, avoiding layout rework.

### 1.3.2 Substrate Connection

Another critical parameter for matching nanoscale devices is the substrate connection,

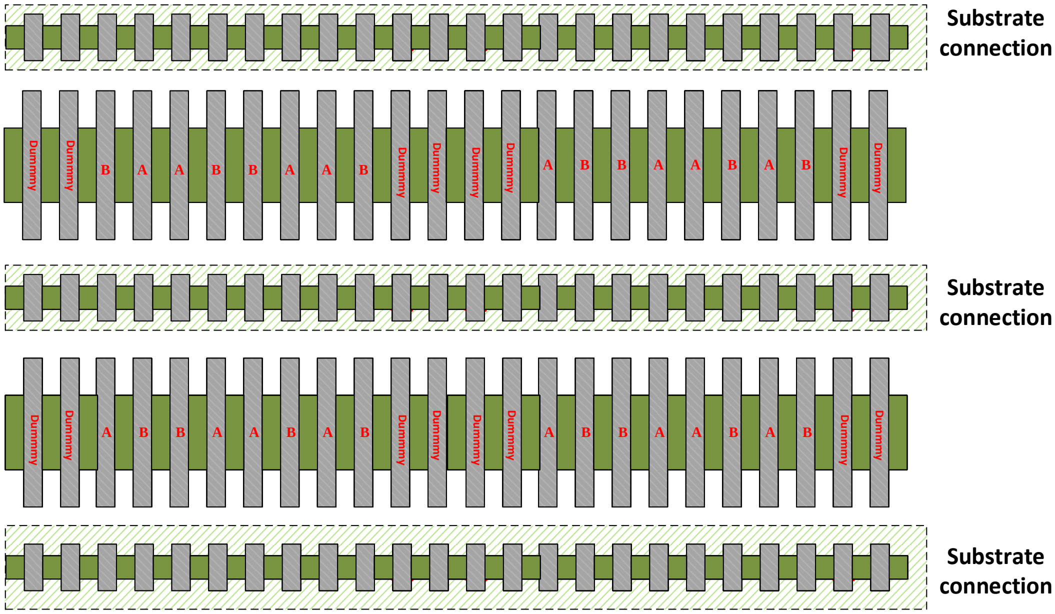


Fig. 1.7 Example of substrate connection in a differential pair.

since matching between closely spaced pairs of devices is influenced primarily by local random and systematic error factors. Matching of devices such as precise capacitors in a large capacitor-DAC (C-DAC) array can be affected by both local and systematic long-distance effects from lithographic patterning several tens of micrometers outside the array, regardless of the technology node. Physical design approaches intended to mitigate these effects in nanoscale technologies are commonly referred as litho-friendly design connection to the substrate [Lewyn-09].

Because nanoscale analog devices are capable of operating at frequencies beyond the 10 GHz range, the location of the substrate ties is a concern in matching differential pairs used in High Frequency (HF) amplifiers. Good device matching and operation at HF requires distance to the well or substrate ties that are uniform and near the source-channel boundary [Lewyn-09].

In the cases described above, substrate connection is another parameter to take in consideration: the location, the size and the wiring of the substrate connection are significant, especially when there are multiple rows in the transistor array. The connection to the substrate should be perfectly matched between the two devices and should consider any possible leakage current. In Fig. 1.7 we can see an example of insertion of substrate connection.

In the following sections, we discuss the two structures under study: the differential pair and stacked devices, as we describe different options for their layouts implementations.

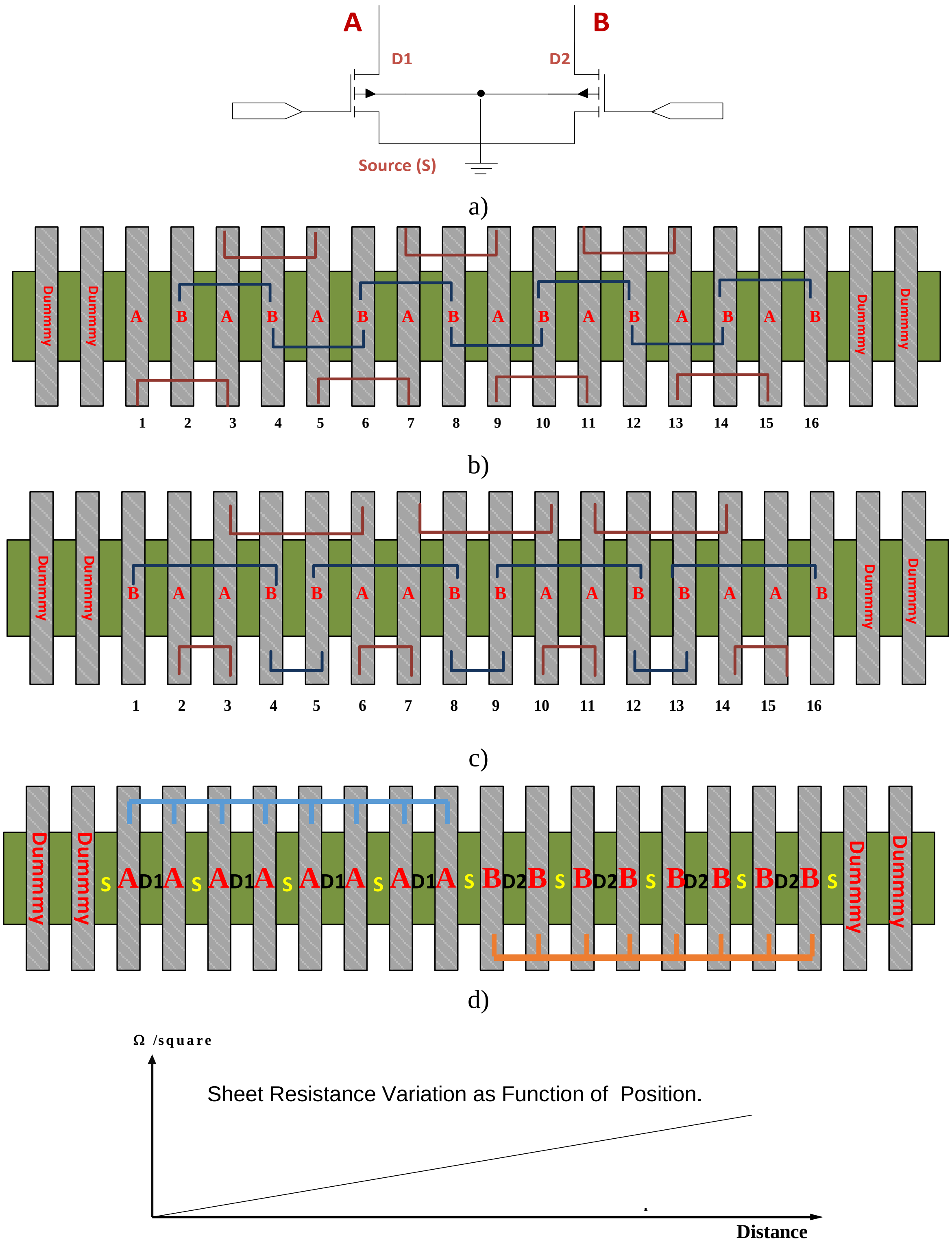
## 1.4. Differential Pair Structures

The differential pair is shown in Fig. 1.8a. This is one of the most common and critical structures used in different kinds of analog circuits. For this circuit, there are many different layout topologies with different grades of complexity; all of them intended to achieve the best possible matching. Some of the factors that we can compare are the shape of the array of transistors: rectangular or square, which are associated with the number of rows used in the circuit layout. Another aspect that we can take into account is the number of dummy devices and its location: either in the middle or at the fringes of the group; and finally, the symmetry implemented in the transistor array. Next, we describe some possible topologies for the layout implementation of a differential pair

One Shared Diffusion (OSD): This topology is shown in Fig. 1.8b. Here the transistors are divided in fingers and placed one next to each other: all the fingers of the transistor A are placed next to all the fingers of transistor B. Since the transistors share the source connection, they always can be placed together, sharing the diffusion associated to this net. Hence, the number of fingers of the transistors can be an odd or an even number. This topology does not have good matching properties; however, it offers a simple interconnection between the nets.

Interdigitated Layout implementation (IL): Presented in Fig. 1.8c, in this topology, the transistors are also divided in fingers, but in this case, they are placed alternating two fingers of transistor A with two fingers of transistors B. Similarly to the OSD topology, the transistors share the diffusion associated to the source net. Since the fingers of the transistors are placed in pairs, the total number of fingers should be an even number.

Common Centroid (CC) layout implementation: This topology is shown in Fig. 1.8d, it is a variation of the interdigitated layout implementation to improve the gradient tolerance in the horizontal direction. This specific implementation is a unidimensional common centroid implementation, since it is only implemented in one row, however, its main characteristic is its symmetry in the y-direction, even though it has the inconvenient of an asymmetric output load. As in the previous case, the transistors share the diffusion associated with the source net, and the number of fingers should be an even number.


A
B
D1
D2
Source (S)
Dummmy
A B A B A B A B A B A B A B A B
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
B A A B B A A B B A A B B A A B
S A D1 A S A D1 A S A D1 A S A D1 A S B D2 B S B D2 B S B D2 B S B D2 B S
Ω /square
Sheet Resistance Variation as Function of Position.
Distance

a)

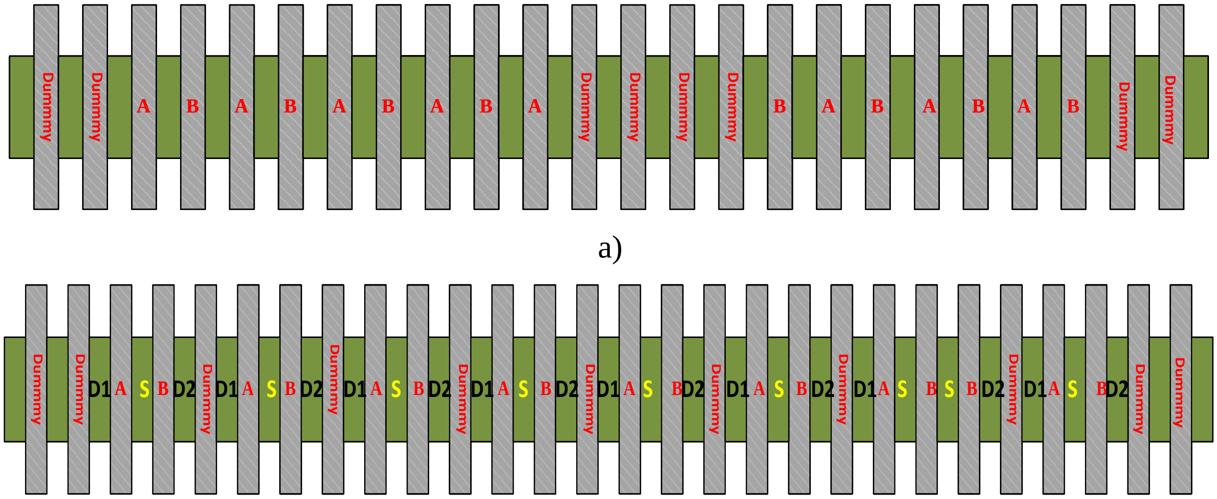
b)

c)

d)

e)

Fig. 1.8 Differential pair layout topologies: a) schematic diagram; b) one shared diffusion; c) interdigitated; d) common centroid; e) variation (gradient) of the square resistance with respect to the distance.


Dummmy Dummmy A B A B A B A B A Dummmy Dummmy Dummmy Dummmy B A B A B A B Dummmy Dummmy
Dummmy Dummmy D1 A S B D2 Dummmy D1 A S B D2 Dummmy D1 A S B D2 Dummmy D1 A S B D2 Dummmy D1 A S B D2 Dummmy D1 A S B D2 Dummmy D1 A S B S B D2 Dummmy D1 A S B D2 Dummmy Dummmy

a)

b)

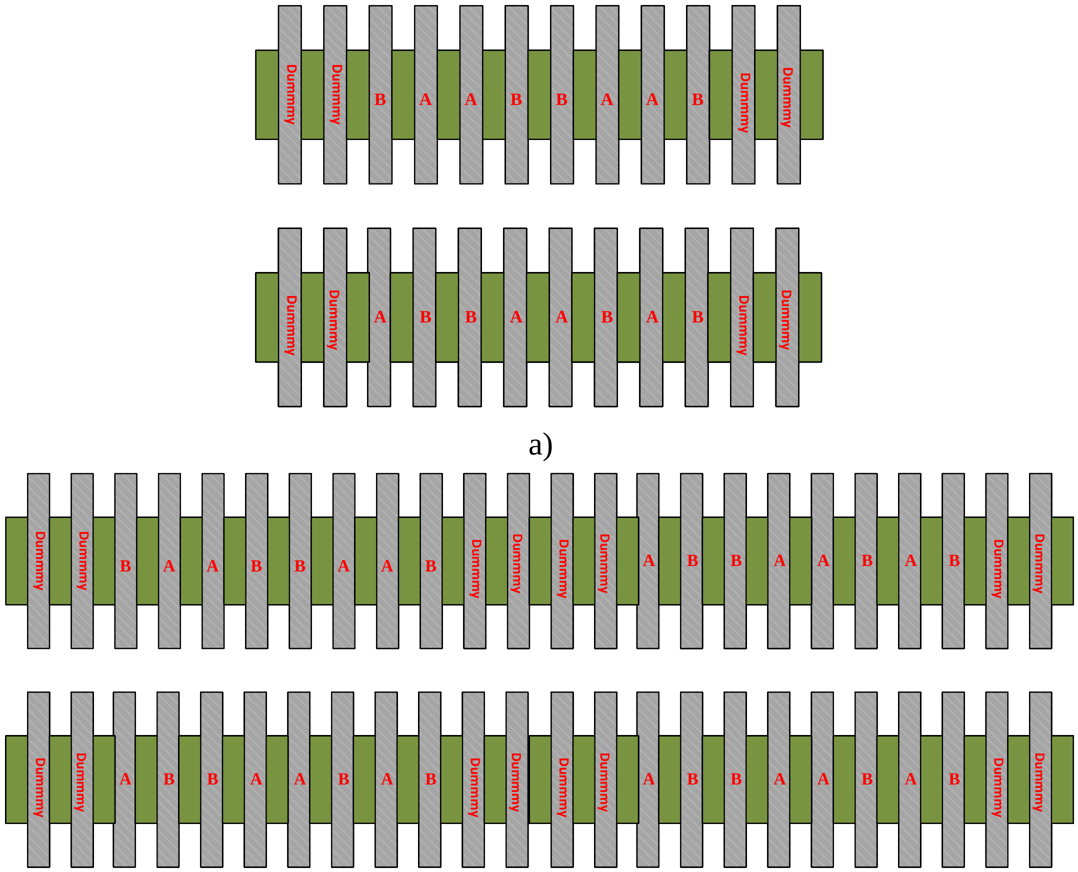
Fig. 1.9 Variations of the location of the dummy devices on a layout implementation: a) dummy insertion in the middle of the array, b) interdigitated layout implementation with dummies.

To illustrate the differences between the three topologies, Fig. 1.8e shows a plot of the variation (gradient) of the square resistance with respect to the distance[2]. Even though the resistance is better balanced in the CC implementation than in the other two topologies as it can be noticed in the connections, the current density is not, since the array has the same transistor at the ends of the array, and the diffusion of this transistor only has half of the current of the other diffusions. As it was addressed the OSD implementation has a poor matching, since all possible variations have a more significant effect in one of the devices (Devices B in Fig. 1.8b), that in the other.

Another possible variable that we can add, in the layout implementation of a differential pair structure is the location of the dummy devices used in the layout. For instance, in all the implementations in Fig. 1.8 the dummy transistors are placed at the sides of the transistor's array, while in Fig. 1.9a there are dummies placed also in the middle. End dummy devices help to improve the matching in the layout, while the middle dummies are used to have a better heat dispersion[S-Shin-16].

Other option is to place the dummy devices between the active transistors: consider the

[2] This plot is a hypothetical example of a linear variation of the material resistance with respect to the distance; it does not represent a physically measured plot. It is used here to illustrate how layout topologies may impact on these variations.


Dummmy Dummmy B A A B B A A B Dummmy Dummmy
Dummmy Dummmy A B B A A B A B Dummmy Dummmy
Dummmy Dummmy B A A B B A A B Dummmy Dummmy Dummmy Dummmy A B B A A B A B Dummmy Dummmy
Dummmy Dummmy A B B A A B A B Dummmy Dummmy Dummmy Dummmy A B B A A B A B Dummmy Dummmy

a)

b)

Fig. 1.10 Variations on layout implementations: a) two sets (two rows) of transistors array to add gradient tolerance, b) common centroid of interdigitated groups.

interdigitated layout with dummy insertion shown in Fig. 1.9b: In the previous topologies, the transistors share the diffusion on the source net; this increases the capacitance associated to this node since its diffusion area in the layout is larger than the diffusion area of the other nets; it also restricts the number of fingers to an even number. In the layout of the Fig. 1.9b, the dummy devices have been added, which reduces the diffusion area of the source (we can see that the geometries at the sides of the array are the diffusions of the drain nets instead of the diffusions of the source) and allows having an odd number of fingers. The addition of dummies can reduce the self-heating effect since it reduces the interaction between the devices and yields more area to dissipate the heat [Shin-16]. The trade-off for this implementation is a significant increase in the area for the entire layout.

Previous layout structures were implemented using only one row for the arrays, but it is also common to have multiple rows of transistors to add gradient tolerance, as shown in 1.10a, or having a common centroid of interdigitated groups to get a better current matching, as shown in Fig. 1.10b.

With these few examples, we show some possibilities that exist to implement a common layout, considering just the topology for the transistor array.

However, in these structures, we can observe some common facts for the implementation of analog layouts, which help to reduce the number of layout options but increase the constraints of the design. For any critical device, we should never use less than 2 legs, since we need to have one leg with a current flow from the left to the right and one with a current flow from the right to the left. Also, it is better to have an even number of rows in the common-centroid arrangement. Therefore, we need to have an integer multiple of 4 to create an array.

It is necessary to conduct an in-depth analysis of these topologies, modifying specific layout parameters to compare the different trade-offs involved.

## 1.5. Stacked Devices Structure

Besides their usage to achieve voltage or current attenuation, the implementation of stacked devices is very common in current nanoscale technologies. In many analog applications stacked digital devices are used rather than one single analog device [Saari-16]. In Chapter 2, a detailed analysis of the tradeoffs due to the use of stacked devices in the implementation of analog circuits is described. This analysis is not the primary purpose of this work; however, it is essential to explain the use of this kind of structures in analog designs and the necessity to have the best possible layout implementation. For simplicity the analysis of the differential pair circuit is not presented in this work.

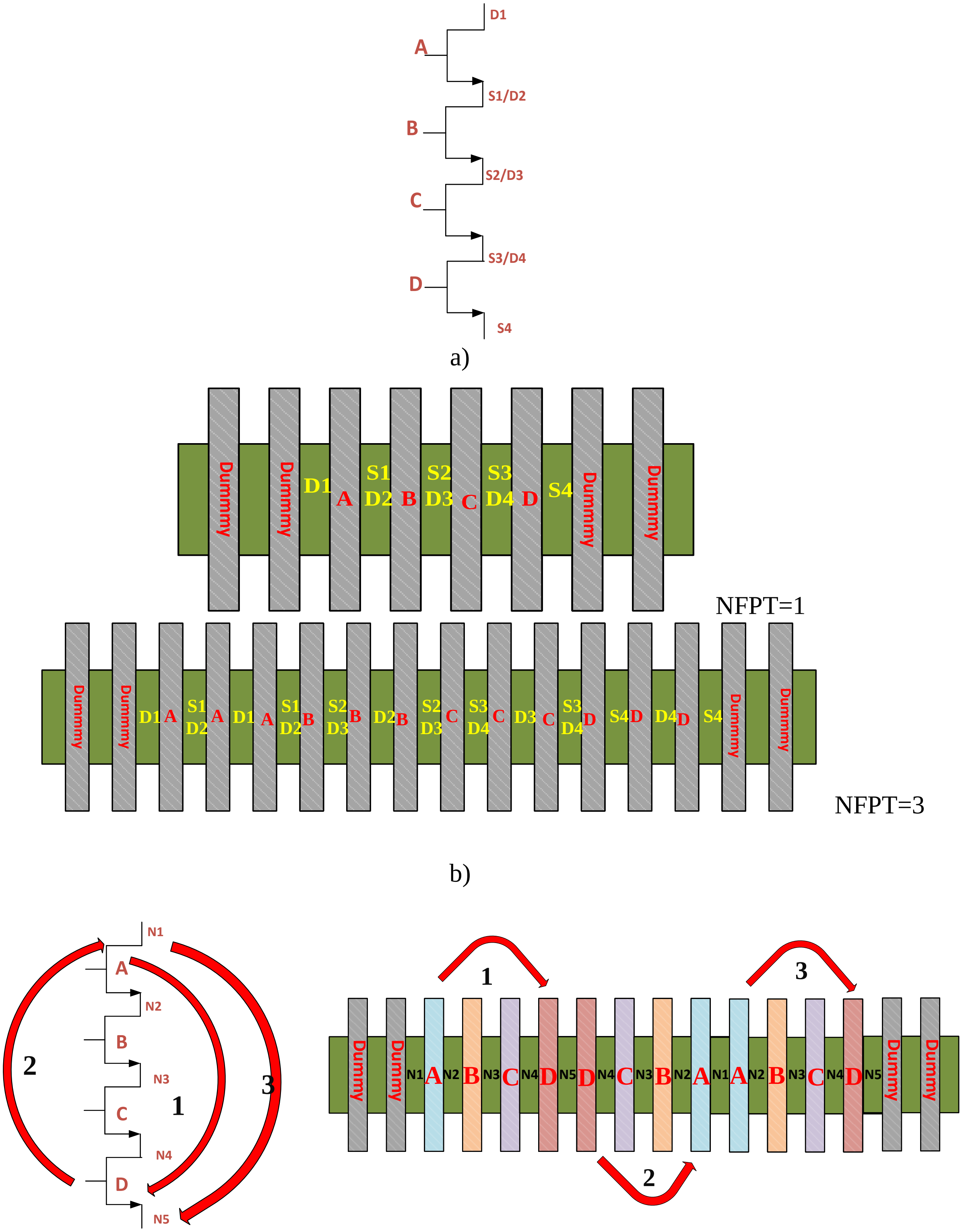
Using, for argument sake, an array of four stacked devices whose schematic diagram is shown in Fig. 1.11a. Some of the possible variants for its layout implementation are the next.

The One Shared Diffusion implementation (Fig. 1.11b). Similarly to the OSD for the differential pair, the transistors are divided into fingers and are placed one next to each other. In the case of the stacked devices, the source of one transistor is shared with the diffusion of the next one; when the layout is implemented, the transistors can share the diffusions associated to these nets (only one diffusion is shared between two transistors). To make possible sharing these diffusions, the number of fingers of each transistor must be an odd number. In Fig. 1.11b there are two examples of this topology: for the number of fingers per transistor (NFPT) is one and when NFPT is three.

The Interdigitated Layout (IL) implementation shown in Fig. 1.11c is used when transistors are divided into at least two fingers; one finger of each transistor is placed next to each other, starting from that one on the "top" of the array and continuing until one finger of the transistor that is on the "bottom"; then the order in which the transistor fingers is placed is inverted. Considering the example in Fig. 1.2c, each transistor is divided into three fingers, a single finger of each transistor is placed followed by a finger of the next one, starting from transistor A and continuing until one finger of the transistor D is placed. Then the process is repeated, but the order of the transistor fingers is inverted, starting from device D to device A; finally, the process is repeated on last time. In this case, the order for the devices fingers is, A-B-C-D-D-C-B-A-B-C-D. Since a single finger is placed at a time, the number of fingers by which all transistors are divided could be an odd or an even number. This topology has a proper matching regarding process gradients, but the heating effects due to current flow are not well balanced; transistor D has the biggest heating effect due to these currents. Also in this topology, there are many routing and connections between the different transistors, the current is continuously "jumping" from one layer to another, which could cause the temperature on the devices to increase (this is described in the next section).

As in the case of the differential pair, it is possible to add dummy devices between the different active devices. In Fig. 1.12 it is shown an interdigitated layout with dummy insertion. In OSD topologies there can be only an odd number of fingers for the same transistor; if an even number is placed, it is not possible to share diffusion with other devices. If it is required to implement the layout with an even number of transistor fingers, then it is necessary to use dummy devices between the different transistors, as shown in Fig. 1.12 (two fingers of transistor A, two fingers of transistor B and a dummy device between them). This allows having an even number of transistor fingers an also helps to reduce the self-heating effect since the area to dissipate heat increases. As in the differential pair topologies, it is possible to place a certain number of dummy devices at the ends or in the middle of the array of stacked devices.

Again we can see that even for a typical structure there are multiple options for its layout implementation, each of them with numerous trade-offs: routing, noise, dispersion, matching, among others. The optimal layout implementation depends on the target application of the circuit, the parameters and effects we need to optimize.



a)

b)

c)

Fig. 1.11 Layout topologies for an array of four stacked devices with a different number of fingers per transistor (NFPT): a) schematic diagram; b) one shared diffusion, two cases are shown, NFPT=1 and NFPT=3; c) interdigitated layout, NFPT=3.

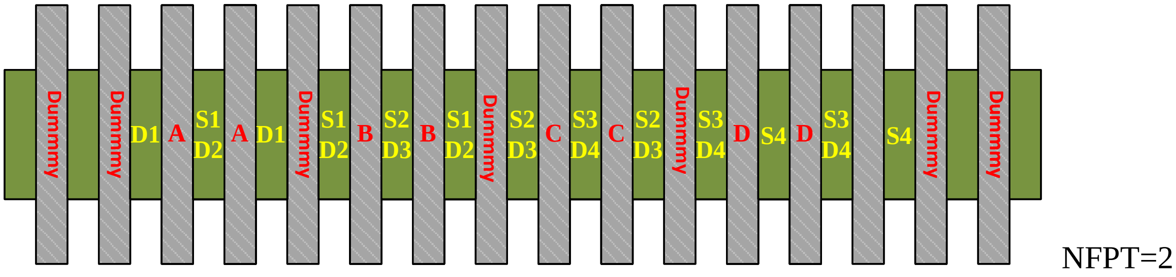


Fig. 1.12 Interdigitated Layout implementation of an array of stacked devices with dummy insertion, NFPT=2.

## 1.6. Device Metallization and Routing

Other factors affecting the performance of a specific layout are the routing and the interconnections of the signals, their location and the way they are routed.

Current technologies are more restrictive for metal interconnection, limiting the range of values for metal's width and space and especially metal's orientation. However, in some technologies, the width of the metal layers and the space between them are not limited to an absolute value, but to a range of them. Furthermore, the designer can modify some of the routing parameters, such as the metal layer to use, the location of the routing and the type and size of the connections (vias). With all these variables, the designers have several options for the routing and interconnection of the circuit’s signals; this is illustrated in Fig. 1.13.

We should remember that resistance and capacitance depend on the metal layer we use. The "higher" the metal layer is, the more stack vias are required for the connection, increasing the resistance seen by the signals. The width of the wires is based on many factors including IR drop, electromigration rules, parasitic resistance and parasitic capacitance (both critical for high-frequency designs) and inductance (significant for RF designs and critical ground loops).

Designers should consider all these variants to select the appropriate topology for the implementation of their layouts; in many cases the chosen topology requires some extra metallization that may impact the performance of the circuit. In the following section, as an example, the routing of two different topologies on an array of stacked devices is described.

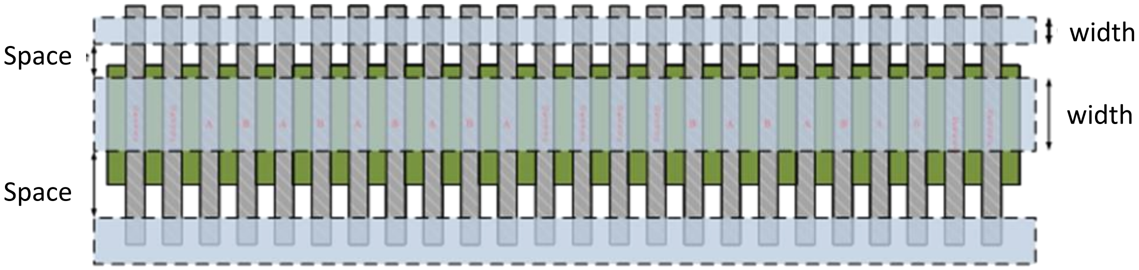


Fig. 1.13 Example of metallization over a differential pair.

## 1.7. Routing in an Array of Stacked Devices

In this section, the layout implementations of the One-Shared Diffusion and the Interdigitated Layout topologies for an array of stacked devices are analyzed; for argument sake, we consider an array of four-stacked devices. If the number of fingers per transistor is one, both layout implementations would be equal; including their routing , this is illustrated in Fig. 1.14.

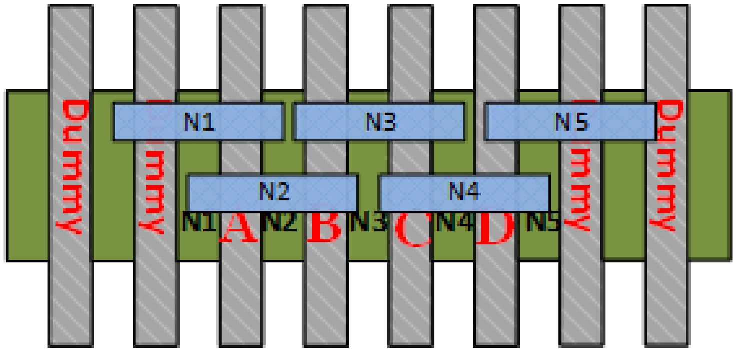
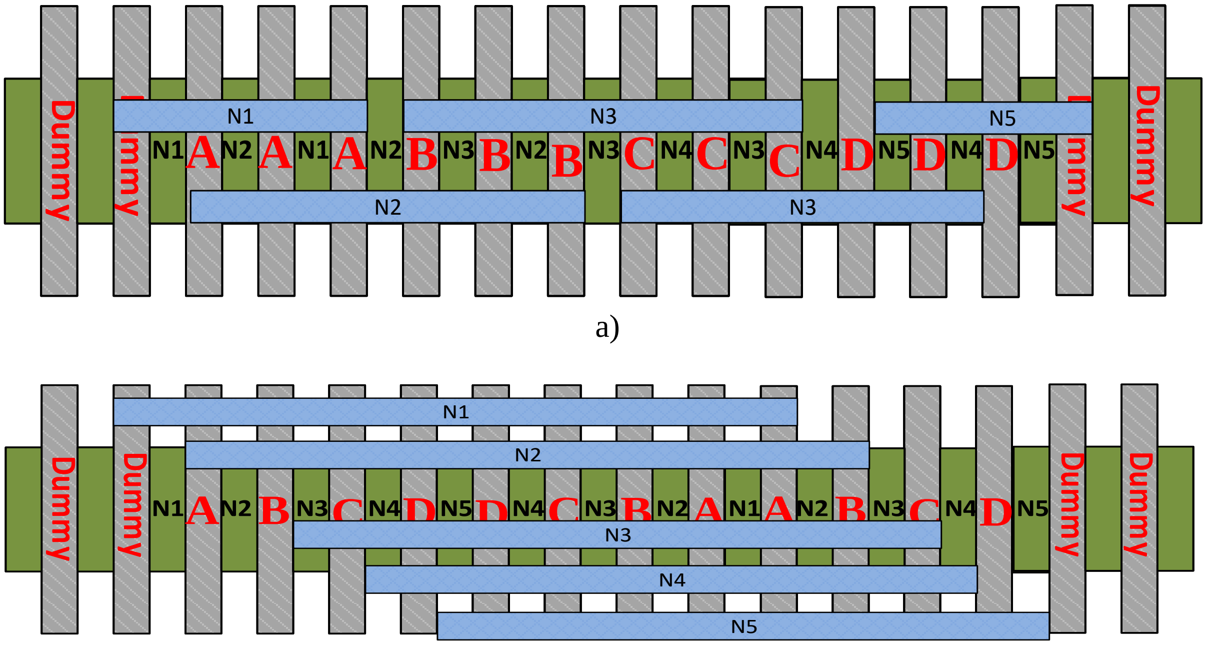


Fig. 1.14 Layout and array of four stacked devices, when the total number of fingers is one.

The interconnection metals shown in Figs. 1.14 to 1.16 should be high enough to not affect the performance of the transistors. In current nanoscale technologies, these interconnection metals normally start at M2.

When the number of fingers is larger than two, the routing used in each topology is different: the lengths of the interconnections are longer in the case of the interdigitated layout topology, as is observed in Fig. 1.15a and Fig. 1.15b. Furthermore, if the number of fingers per transistor increases, the difference between the interconnections length also increases. Similarly,

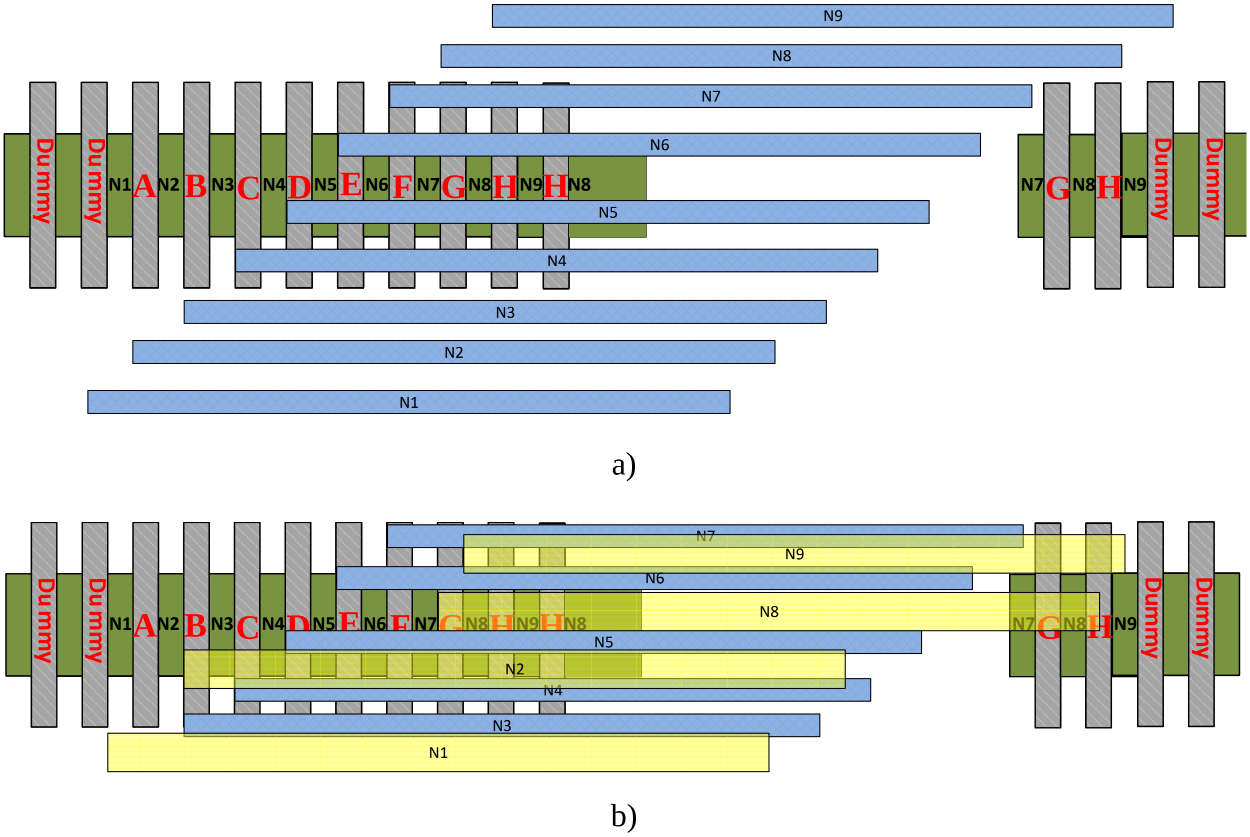

N1
N3
N5
Dummy
mmy
N1 A N2 A N1 A N2 B N3 B N2 B N3 C N4 C N3 C N4 D N5 D N4 D N5
mmy
Dummy
N2
N3
N1
N2
Dummy
Dummy
N1 A N2 B N3 C N4 D N5 D N4 C N3 B N2 A N1 A N2 B N3 C N4 D N5
Dummy
Dummy
N3
N4
N5

a)

b)

Fig. 1.15 Comparison between the routing lengths for both topologies: a) one shared diffusion; b) interdigitated layout. The figure is taken from [Lomeli-Illecas-17].

if more transistors are added to the array, the length of the interconnections on the interdigitated layout topology are much longer that one of the shared diffusion topology; besides, more routing tracks are required; this is also illustrated in Fig. 1.14, where we can also see that in the case of One Shared Diffusion topology, the metals used for routing can share tracks, while in the case of the Interdigitated Layout topology each metal layer needs an individual routing track. As more transistors are added to the array, more routing tracks are required, increasing the required area for the corresponding layout implementation. One solution is to route some of the signals using higher metal layers, as it is illustrated in Fig. 1.16; this solution reduces the required area for the layout implementation. The disadvantage of this approach is the complexity of the layout design since more elements need to be added to the implementation such metal layers and vias/contacts. Also, another drawback is that the mismatch between the interconnections increases since they are formed, now by different elements.



a)

b)

Fig. 1.16 Examples of routing for an interdigitated layout topology: a) using the same metal; b) using different metals.

## 1.8. Design Rules

The layout implementations presented in previous sections are exposed to the limitations and restrictions of the design rules (DR) of the target fabrication technology. In current layout technologies, the design rules are defined by discrete values (a pitch), as in the case of the transistor's width. Other examples include the space between diffusions (of the same or different type), as shown in Fig. 1.17a and the distance between diffusion and the "N-well," Fig. 1.17 b. These technology rules can affect some transistor parameters, for instance, $V_t$ could be changed by the proximity of the gate to the N-well edge.

DR can also be limited within a range of continuous values. A minimal and a maximal limit bounds the length of the transistor's end cap; end cap is the "extra" poly wire, which does not cover transistor diffusion, as shown in Fig. 1.17c.

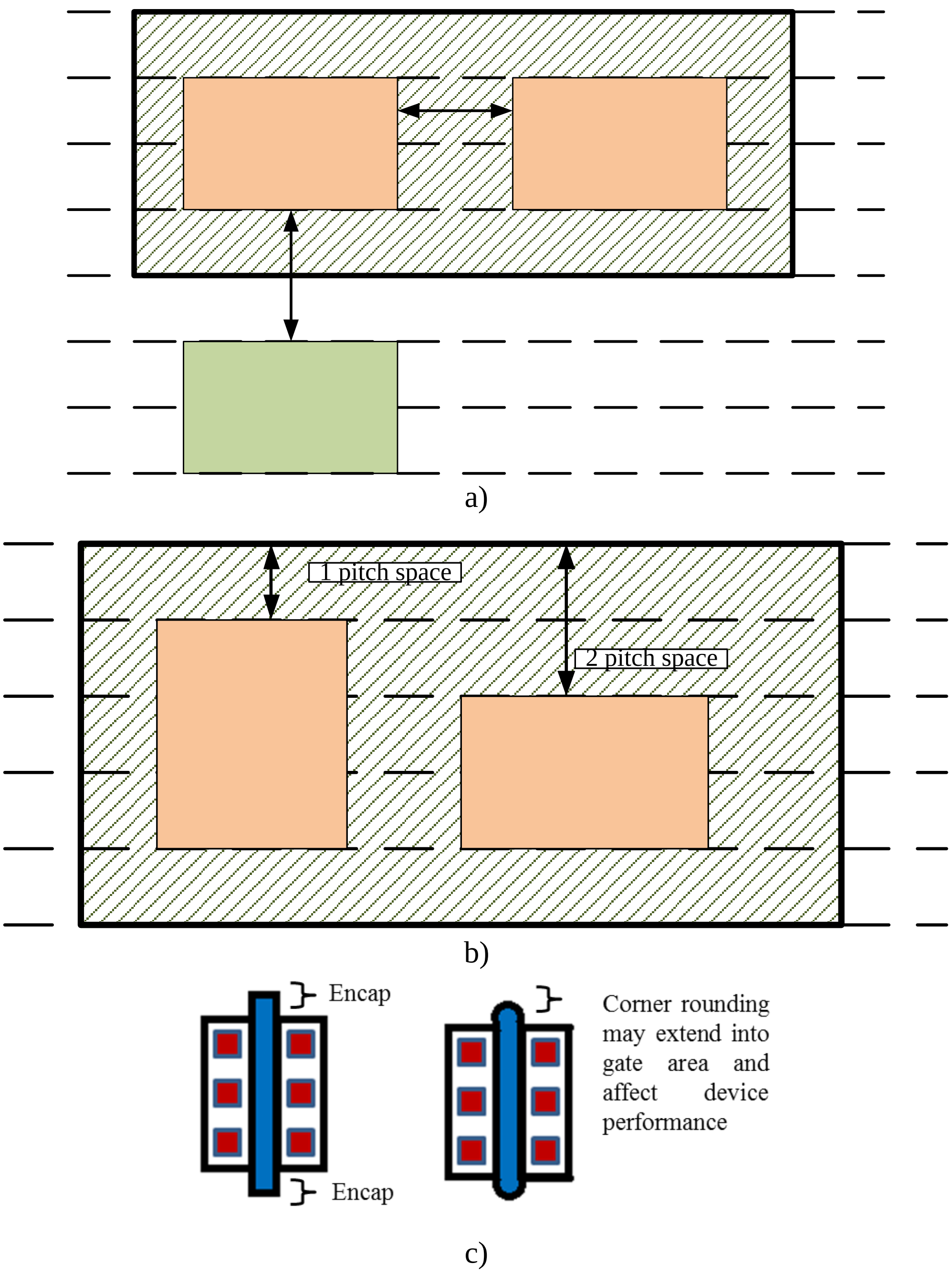

a)
1 pitch space
2 pitch space
b)
Encap
Encap
Corner rounding may extend into gate area and affect device performance
c)

Fig. 1.17 Examples of dimensions that are defined by technology DR: a) space between diffusions b) space from diffusion to n-well c) endcap length.

The election for the value of a specific design rule, applied to one of the topologies previously described, could significantly affect its correct functionality. An in-depth analysis of the circuit performance for a specific design rule could help to optimize layout the implementation of the circuit.

## 1.9. Proposal

In previous sections we described different options for the layout implementation of the differential pair and arrays of stacked transistors. For these structures there are a large number of options for their layouts, not only for their topology or placement, but also for the interconnections and the design rules related to them. Similarly, the layout implementation of analog circuits in modern nanoscale technologies is a task that requires in depth understanding of non-desired physical effects and knowledge of layout design techniques to compensate them, which leads to long design cycles and high design costs. Then, to reduce design cycles and cost it may be necessary to have a design tool that helps to account for those non-desired effects. This tool should allow the analysis of different layout topologies and the selection of the optimal layout implementation for each particular application in a short time.

As it was addressed before, many efforts have been made to create an automatically generated analog layout. A general desired objective is to unify these efforts to enable a tool that allows quick layout implementation and its extraction to come up with an optimal analog circuit.

In addition to the approaches addressed previously, some other techniques for the layout placement generation are the following:

In the proposal presented in [Saravanan-11], groups of cells are organized in arrays, which are later connected together through metal/via connections. In [Jangkrajarng-03], using insertion techniques an automatic active device layout generation tool is enhanced by adding the capability to insert analog layout modules. The device width, length, and finger variables are handle by a device layout generator with that deploys a design-space exploration engine.

Another approach based on constrains definition technique generates hierarchical placement rules based on a netlist, a building blocks library and symmetry analysis to determine the constraints [Yilmaz-09]. Considering the priority of the constraints, a tiered partitioning of the circuit including matching, proximity and symmetry groups is computed automatically and furthered to a tool for placement purposes [Subramaniam-12].

In general, all analog layout automation approaches consist of three main processes, namely: Placement Generator, Matching or Array Alignment, and Auto Routing [Saravanan-11].

The proposal presented in this work is based on the placement generation of the devices, through the creation of layout templates. These templates guide the placement of the layout devices using different topologies and configurations. The matching and the routing of the layout is determined by the templates.

In this thesis, we present the development of a CAD tool that accelerates the layout implementation of the analog circuits described in this chapter. The use of this tool facilitates to circuit designers the analysis, characterization, and optimization of their designs. The information provided by the tool helps to understand how the non-idealities of the layout could affect the performance of the circuits and then the selection of more robust and reliable layout structures.

The proposed CAD tool is focused on structures formed exclusively by transistors. Passive structures, such as capacitors, inductors or resistors, are beyond the scope of this work, since in most of the current nanoscale technologies this kind of passive circuits are created based on predefined templates included in the technology library. The options for designers are usually limited to the placement of these templates, whose guidelines are also included in the process technology used.

The tool generates multiple layout configurations of the same circuit with different levels of detail, which can be used to create a layout database or library. Based on a simplified parasitic extraction process, designers can analyze the trade-offs between the different layout versions to optimize the design and then select the layout version that better fits their needs. Usually, the traditional verification or validation tools take long time and are executed at the very end of the design process. Performing this simplified extraction process in the early stages of the design process helps to avoid multiple iterations and re-work. In the proposed CAD tool, designers are able to define layout requirements such as, number of dummy devices, number of rows where the layout should be implemented, among others and then the tool takes that information to generate different layout views for the differential pair and arrays of stacked transistors, with its corresponding characteristics and saved in a layout database.

## 1.10. Summary

In this chapter, we reviewed the current challenges to design in new fabrication technologies starting from the 22nm node, as well as, the efforts that designers have to do to adapt their implementations to layout limitations in these technologies, which include: dimensions discretization, and increment of the leakage current, among others. As it was commented, in the next chapter a detailed analysis of the use of arrays of stacked devices for the implementation of analog circuits is presented; this study illustrates the importance of this kind of circuit to overcome some of the challenges faced in current nanoscale technologies.

In this chapter 1 we have also seen that even for common and well-known structures there are multiple options for their layout implementation, each of them with numerous trade-offs, being matching, metal routing and its surrounding environment among the most important.

It is necessary to have an efficient and accurate method to analyze the non-idealities, to select and implement the best layout option. The proposed solution consists in the development of a tool that accelerates the layout implementation of selected analog circuits, for their subsequent analysis, characterization, and optimization. The fast implementation of these layout versions and the generation of the corresponding libraries for the study of their limitations and tradeoffs may help designers in the implementation of their circuits.

In Chapter 3 the development of the analog layout synthesis tool is presented, including a detailed description of the process followed by the tool, as well as the description of the script that forms it, including their inputs, expected outputs, their correlation and possible improvements.

# 2. Analysis of the Implications of Using Stacked Devices in Analog Circuits Implemented with Nanoscale Technologies

In this Chapter a complete analysis of the implications of using arrays of stacked devices in analog circuits implemented with nanoscale technologies is conducted. In previous chapter, the limitations, in current nanoscale technology processes, for the implementation of analog circuits, due the used of discretized devices are addressed. To mitigate these limitations, and also the reduction of the dimensions of these devices, and the reduction of power supply voltages, the designers, are implementing new structures or topologies capable of emulating the correct analog behavior [L.-Lewyn 09]. One of the solutions consists of using transistors arranged in stack [Saari-16].

A detailed analysis of this structure considering different layout implementations is presented. The results of the analysis illustrate the advantages and importance of the use of array stacked devices for the implementation of analog circuits in current nanoscale technologies. The analysis also illustrates the importance to have a tool that allows fast layout implantation of this kind of structures, due to the vast number of options that exist for their layout implementation, as is addressed in the different chapters of this work.

This analysis considers the simulation of the circuit's schematics, where the effects of using arrays of stacked transistors in substitution of a single device are deeply analyzed. These effects are first studied by obtaining the characteristics curves of the devices to illustrate the interrelation between the input voltage and the output current of the transistor arrays. Parameters studied are the output resistance of the devices, the channel length modulation factor, the leakage current and the propagation delay time. Additionally, we review these effects by comparing the responses of a current mirror circuit implemented with stacked transistors and when it is implemented using single devices. The information presented in this chapter is described in more detail in [Lomelí-Illescas-17].

## 2.1. Effects on I/V Characteristics

A fundamental limitation for the implementation of analog circuits lies in the lack of flexibility to define the transistors' width and length. The width is limited to a set of discrete values, while transistor's length is normally limited to one single fixed value. As it is well known, the value of the transistor's drain current in linear and saturation region can be approximated by:

$$I_d = \mu C_{ox}\frac{W}{L}\left\{(V_{GS}-V_T)V_{DS}-\frac{{V_{DS}}^2}{2}\right\} \tag{2-1}$$

$$I_d = \mu C_{ox}\frac{W}{L}(V_{GS}-V_T)^2 \tag{2-2}$$

It is clear from (2-1) and (2-2) that drain current value, *I<sub>d</sub>*, is proportional to the gate to source voltage (*V<sub>GS</sub>*) and the ratio between the transistor's width (*W*) and length (*L*).

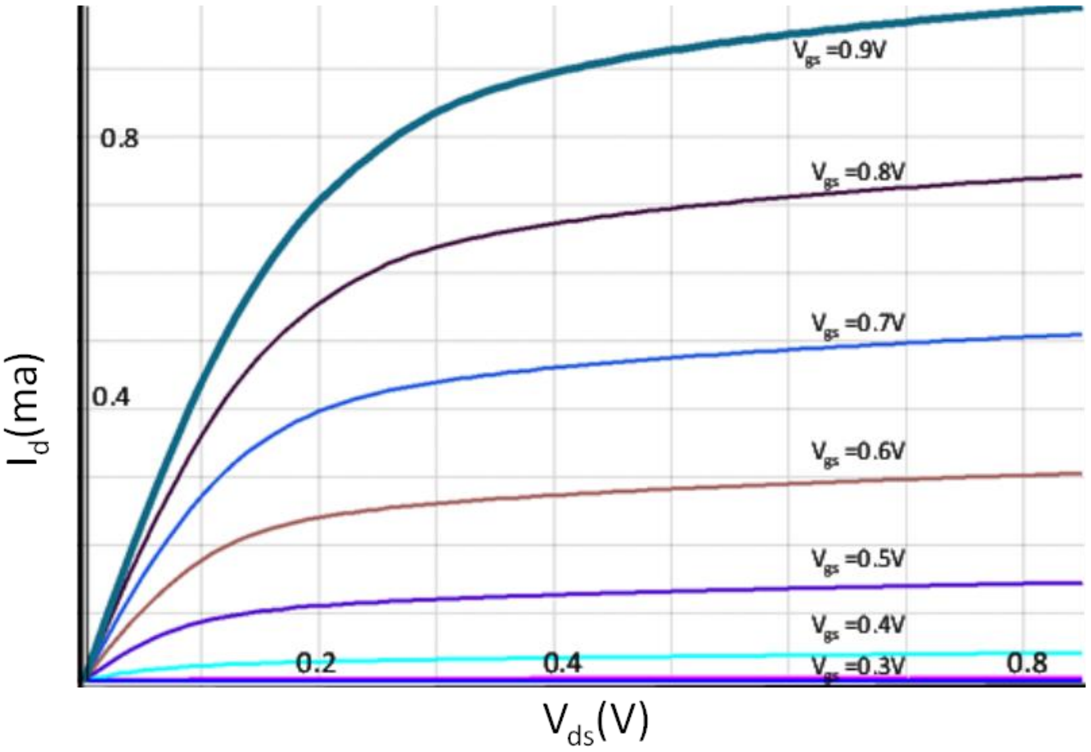


Fig. 2.1 Characteristics curves of an NMOS transistor for a given *W/L* value and multiple values of $V_{GS}$.

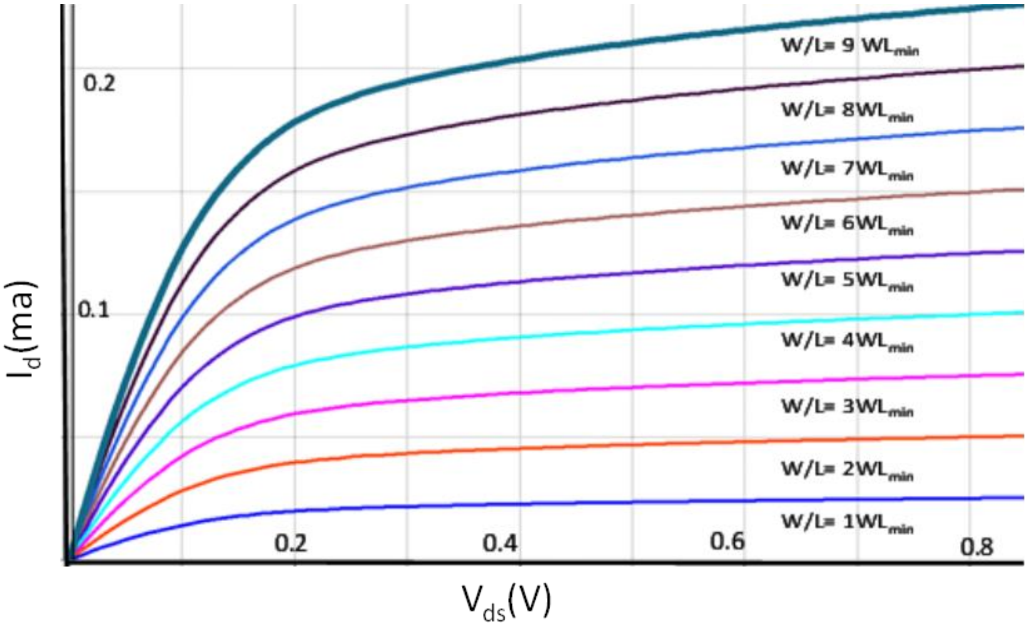


Fig. 2.2 Characteristics curves of an NMOS transistor for a given $V_{GS}$ value and multiple values of W/L.

In Fig. 2.1, the characteristics curves of an NMOS transistor, for specific width and length sizes and for different $V_{GS}$ values, are presented, while in Fig. 2.2 are shown for a specific $V_{GS}$ value and different width sizes. Let $W_{min}$ denote the minimum feasible transistor width and $WL_{min}$ the ratio between $W_{min}$ and the fixed $L$ associated with the technology. Based on this, for a specific bias condition, the $I_d$ value will be a multiple of $WL_{min}$. As it was previously addressed, the options for the design values of this variable are very limited. One of the solutions to increase the number of values available for the designs is the use of transistor placed in stack. A stacked array of $N$ transistors is equivalent to a single transistor with $N$ times its length [Kong,-16]. With this, it is possible to change not only the value of the transistor's width but also its length, or at least emulate this variation. The options for the selection of the width and length sizes are still discrete; however, this still increases the number of design options to achieve the desired performance for analog circuits. This is illustrated in Fig. 2.3, where the characteristic curves of transistors are shown for multiple possibilities of width and length (multiple numbers of stacked devices) values. From this figure, we observe that using stacked devices increases the number of design options; however, they are still limited to discrete values instead of a series of continues values. A summary of the $I_d$ current values for different choices of $W$ and $L$ ratios is presented in Table 2.1. The ratios of $W$ over $L$ are defined in terms of $WL_{min}$.

TABLE 2.1. SUMMARY OF TRANSISTOR'S $I_{DS}$ CURRENT FOR DIFFERENT OPTIONS OF W AND L VALUES

| | $I_{DS}$ current | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Width (Wmin) | Stack dev=1 | | Stack dev=2 | | Stack dev=3 | | Stack dev=4 | | Stack dev=5 | |
| | W/L | $I_{DS}$ (mA) | W/L | $I_{DS}$ (mA) | W/L | $I_{DS}$ (mA) | W/L | $I_{DS}$ (mA | W/L | $I_{DS}$ (mA |
| 1 | 1.0 | 0.023 | 0.5 | 0.204 | 0.3 | 0.141 | 0.3 | 0.108 | 0.2 | 0.073 |
| 2 | 2.0 | 0.045 | 1.0 | 0.204 | 0.7 | 0.141 | 0.5 | 0.108 | 0.4 | 0.073 |
| 3 | 3.0 | 0.068 | 1.5 | 0.204 | 1.0 | 0.141 | 0.8 | 0.108 | 0.6 | 0.073 |
| 4 | 4.0 | 0.091 | 2.0 | 0.204 | 1.3 | 0.141 | 1.0 | 0.108 | 0.8 | 0.073 |
| 5 | 5.0 | 0.113 | 2.5 | 0.204 | 1.7 | 0.141 | 1.3 | 0.108 | 1.0 | 0.073 |
| 6 | 6.0 | 0.136 | 3.0 | 0.204 | 2.0 | 0.141 | 1.5 | 0.108 | 1.2 | 0.073 |
| 7 | 7.0 | 0.159 | 3.5 | 0.204 | 2.3 | 0.141 | 1.8 | 0.108 | 1.4 | 0.073 |
| 8 | 8.0 | 0.181 | 4.0 | 0.204 | 2.7 | 0.141 | 2.0 | 0.108 | 1.6 | 0.073 |
| 9 | 9.0 | 0.204 | 4.5 | 0.204 | 3.0 | 0.141 | 2.3 | 0.108 | 1.8 | 0.073 |
| 10 | 10.0 | 0.226 | 5.0 | 0.204 | 3.3 | 0.141 | 2.5 | 0.108 | 2.0 | 0.073 |

## 2.2. Channel Length Modulation

Ideally, in saturation region, CMOS transistors should behave as an ideal voltage-controlled current source. For a given $V_{GS}$, $I_d$ should be constant and independent of $V_{DS}$. However, we know that the effective channel length is modulated by $V_{DS}$; an increase in $V_{DS}$ causes the depletion regions and the drain junction to grow, and the length of the effective channel is reduced. A more accurate equation to describe the transistor current in saturation region accounts for the channel length modulation:

$$I_d = I_{Sat}(1 + \lambda V_{DS}) \qquad (2\text{-}3)$$

where $I_{Sat}$ is given by (2-2) and λ is the channel length modulation factor, which is proportional to the inverse of the channel length. This factor typically increases for small devices [Kong-16]. In Fig. 2.4, a comparison of the $I_d$ curves of CMOS transistors for different *W/L* ratios and a different number of stack devices is shown. In this figure we can observe three different groups of curves; for each group, the *W/L* ratio is the same, but the values of *W* and the number of

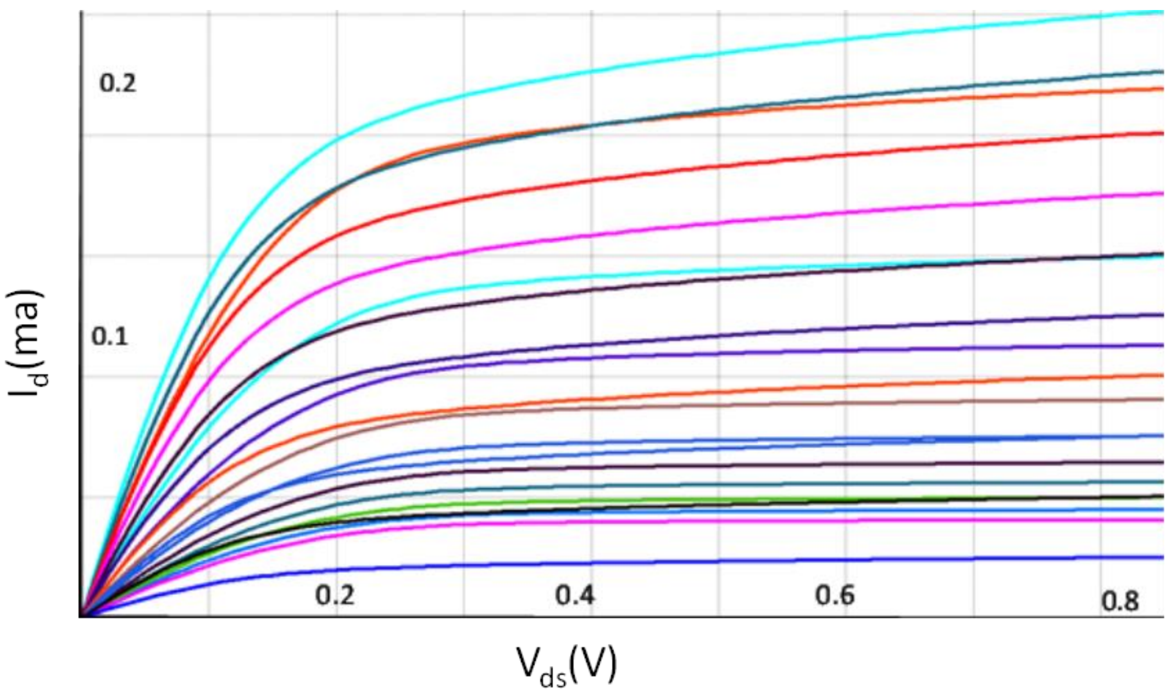


Fig. 2.3 Characteristics curves of an NMOS transistor for multiple widths and lengths values; stacked devices are used to emulate the variation in the length of the device.

stacked devices change. The slope values of these curves in the saturation region are shown in Table 2.2. Since the inverse of this slope represents the output resistance of the transistor, it is seen from Table 2.2 that the effect of the channel-length modulation factor is less important for long channel transistors than for short-channel transistors. In addition, we can notice that $I_d$ for specific bias conditions is higher when the number of stack devices is larger. The use of stack devices helps to obtain a better output resistance and smaller losses in the $I_d$ current. However, the improvement becomes less significant as we continue increasing the number of devices in stack.

TABLE 2.2. SLOPES FOR DIFFERENT W/L RATIOS

| Width ($W_{min}$) | Number of stack devices | $W/L$ ($W/L_{min}$) | $I_{DS}$ (mA) | Slope ($I_{DS}/V_{DS}$ ×$10^{-3}$) |
|---|---|---|---|---|
| 16 | 2 | 8 | 0.2420 | 0.3420 |
| 8 | 1 | 8 | 0.1810 | 0.0458 |
| 16 | 4 | 4 | 0.1077 | 0.0228 |
| 8 | 2 | 4 | 0.1021 | 0.0174 |
| 4 | 1 | 4 | 0.0906 | 0.0122 |
| 16 | 8 | 2 | 0.0545 | 0.0114 |
| 8 | 4 | 2 | 0.0538 | 0.0087 |
| 4 | 2 | 2 | 0.0510 | 0.0061 |
| 2 | 1 | 2 | 0.0453 | 0.0044 |

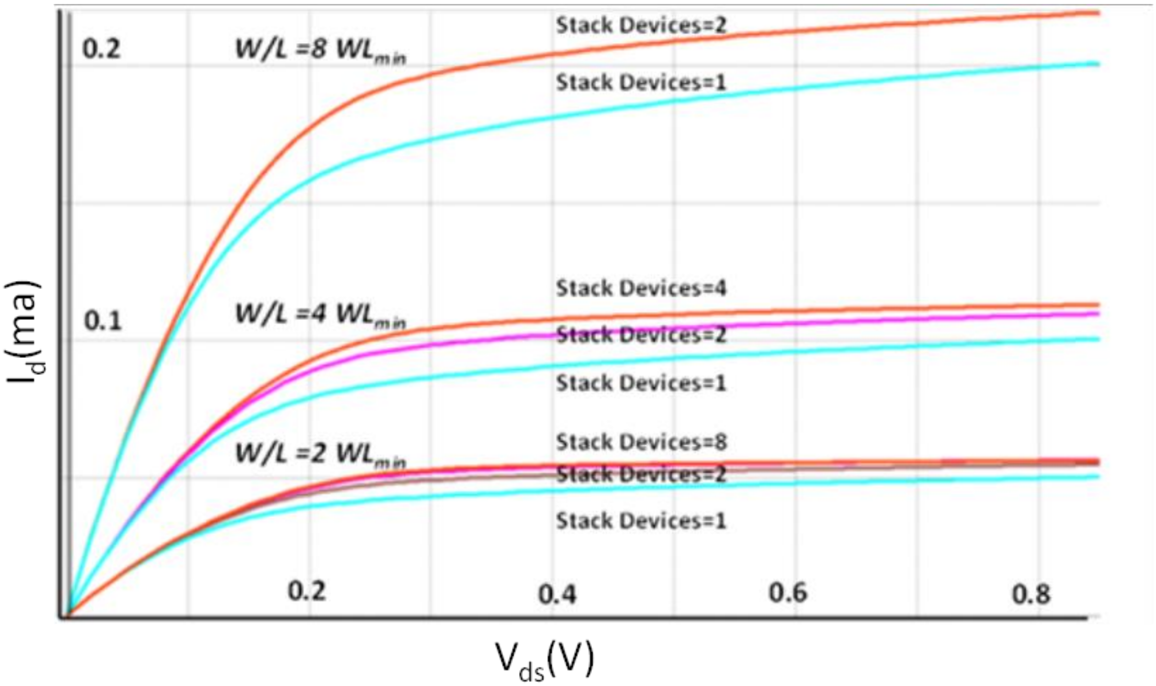


Fig. 2.4 Characteristics curves of an NMOS transistor for multiple *W* and *L* values; in each group the *W/L* ratio is the same, while the values of *W* and the number of stacked devices change.

## 2.3. Leakage

With the continuous scaling of CMOS devices, the leakage current is becoming a significant contributor to the total power consumption in a system. One of the most challenging aspects of today's CMOS VLSI circuits is the standby power dissipation. Feature size reduction has made the effects of leakage currents more pronounced; this becomes more complicated in sub 100 nm technologies with not only subthreshold leakage but also with the gate oxide. Many proposals have been developed to reduce its impact [Saxena-13], [Narendra-01]. In general, stacked devices have smaller leakage than the sum of the leakages consumed by all the devices, individually. This characteristic is often referred to as the stack effect. An informal definition of stack effect is the total leakage current of cascade transistors chain decrease with the number of stacked transistor increasing and is often used to reduce the leakage power [Saxena-13]. In modern sub-micron devices, the threshold voltage may decrease for longer channels due to the reverse short channel effect. Therefore, leakage reduction is less effective, but it is still a commonly used technique. In Fig. 2.5 are shown plots of the transistor current when it is on sub-threshold voltage, for multiple numbers of stack devices. The corresponding results are summarized in Table 2.3. These results illustrate that in off state the subthreshold current is significantly smaller than for a single device.

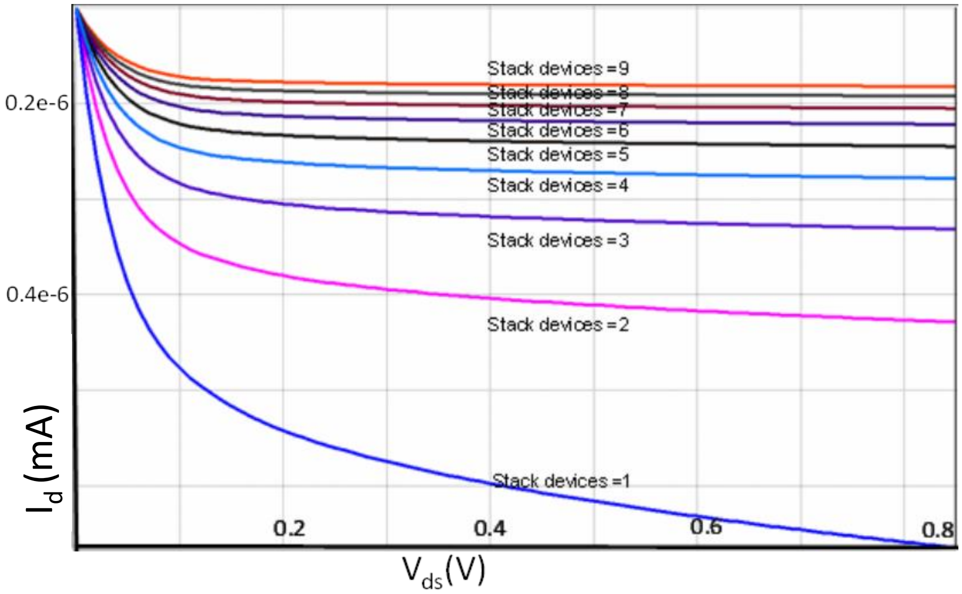


Fig. 2.5 Curves of an NMOS transistor in off state for a different number of stacked devices.

TABLE 2.3. RESULTS OF LEAKAGE CURRENT AND DELAY TIME VARYING THE NUMBER OF STACKED DEVICES

| Number of stack devices | Leakage current (mA) | Delay time (ns) |
|---|---|---|
| 1 | 0.0750 | 0.0063 |
| 2 | 0.0260 | 0.0137 |
| 3 | 0.0085 | 0.0179 |
| 4 | 0.0070 | 0.0214 |
| 5 | 0.0060 | 0.0246 |
| 6 | 0.0052 | 0.0278 |
| 7 | 0.0045 | 0.0309 |
| 8 | 0.0032 | 0.0350 |

## 2.4. Current Mirror

In the previous analysis, we have considered the impact of using stacked devices on the performance of a standalone device. In this section, the impact of using arrays of stacked transistors on a more complex analog implementation is analyzed. The circuit that is studied is a current mirror. Conceptually, an ideal current mirror is an ideal current amplifier. Some important feature of the current mirror are:

g) Relatively high output resistance, which helps to keep the output current constant regardless of load conditions.

h) Relatively low input resistance, which helps to keep the input current constant regardless of drive conditions.

i) Output current $i_o$ linearly related to the input current ii, then $i_o = A_i$, where $A_i$ represents the corresponding current gain.

A basic current mirror is shown in Fig. 2.6. From this figure, if we assume that $V_{DS2} > V_{GS}$-$V_{T2}$, and assuming that the channel length modulation factor is the same for both transistors, $\lambda_1 = \lambda_2 = \lambda$, then $i_o$ can be obtained as

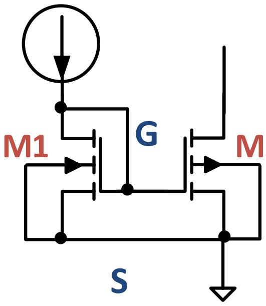


Fig. 2.6 Schematic diagram of a simple current mirror.

$$\frac{i_o}{i_i} = \left(\frac{L_1 W_2}{L_2 W_1}\right)\left(\frac{V_{Gs} - V_{T2}}{V_{Gs} - V_{T1}}\right)^2 \left[\frac{1 - \lambda_{VDS2}}{1 - \lambda_{VDS1}}\right]\left(\frac{K_2}{K_1}\right) \tag{2-4}$$

If the transistors are matched, then $K_1 = K_2$ and $V_{T1} = V_{T2}$, yielding

$$\frac{i_o}{i_i} = \left(\frac{L_1 W_2}{L_2 W_1}\right)\left[\frac{1 - \lambda_{VDS2}}{1 - \lambda_{VDS1}}\right]\left(\frac{K_2}{K_1}\right) \tag{2-5}$$

If $v_{DS1} = v_{DS2}$, then

$$\frac{i_o}{i_i} = \left(\frac{L_1 W_2}{L_2 W_1}\right) \tag{2-6}$$

Therefore, the sources of error are $V_{DS1} \neq \mathrm{V}_{DS2}$ and mismatched between M1 and M2 [Geiger-90].

If the transistors are matched, and the *W/L* ratios are equal but $V_{DS1} \neq \mathrm{V}_{DS2}$, then

$$\frac{i_o}{i_i} = \left[\frac{1 - \lambda_{VDS2}}{1 - \lambda_{VDS1}}\right] \tag{2-7}$$

Since we are assuming that the channel length modulation parameter is the same for both transistors ($\lambda 1 = \lambda 2 = \lambda$ ), we obtain the error plot illustrated in Fig. 2.. We can see that the error directly depends on the value of the channel length modulation, and this value, as indicated before, is proportional to the inverse of the channel length.

In Fig. 2.7, the output resistance of a simple current mirror is shown, varying the number of transistors placed in stack. It is then derived from Fig. 2.7 that the channel modulation effects on the current mirror are reduced when more transistors in stack are used. In Table 2.4, these results including the matching percentage between input and output currents of the circuit, are summarized. From here, it is confirmed that a better current matching is achieved as we increase the number of stacked devices.

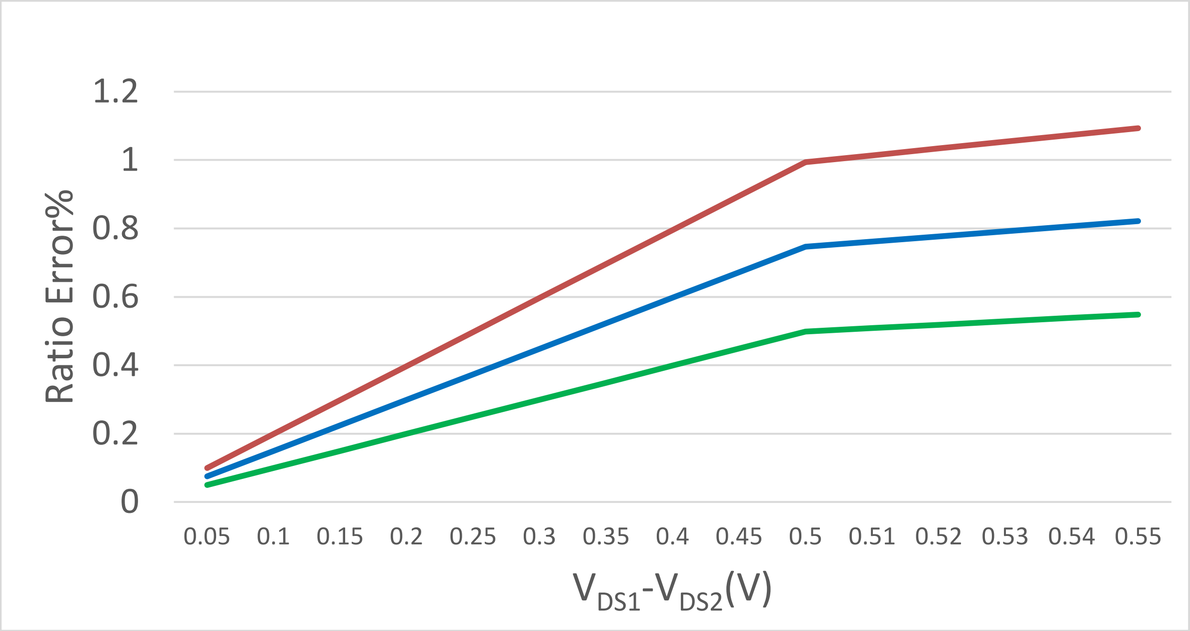


Fig. 2.7 Percentage of error on current matching for different λ values.

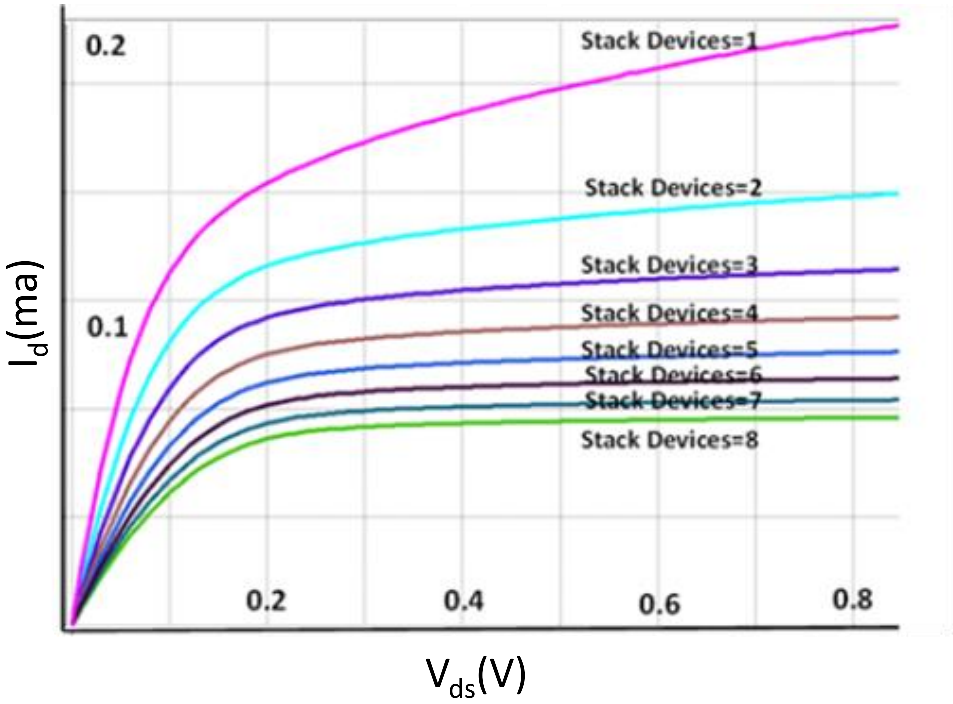


Fig. 2.8 Curves for the output resistance of a simple current mirror using a different number of stack devices

TABLE 2.4. RESULTS FOR CURRENT MIRROR VALUES VARYING THE NUMBER OF STACKED DEVICES

| Number of<br>Stack devices | %<br>matching | Slope<br>($I_{DS}/V_{DS}$ ×$10^{-3}$) |
|---|---|---|
| 1 | 92.094 | 0.00926 |
| 2 | 96.315 | 0.00371 |
| 3 | 97.484 | 0.00220 |
| 4 | 97.840 | 0.00153 |
| 5 | 98.380 | 0.00114 |
| 6 | 99.107 | 0.00131 |
| 7 | 99.310 | 0.00089 |
| 8 | 99.780 | 0.00061 |

## 2.5. Delay Time

We have addressed some of the advantages of using stacked devices; however, some adverse effects need to be considered. There is a tradeoff between power and delay in the propagation of signals. Due to the input load requirement and due to the stacking of devices, the drive current of a forced-stack gate is lower, resulting in an increased delay time [Narendra 01]. The reduction in leakage due to the stack effect can lead to an increase in delay; hence, it can be used in situations where this delay can be tolerated or by using gates with natural stack [Saxena-13]. In Table 2.3, the delay time results for the propagation of a pulse at the input of an inverter that uses a different number of stacked devices are presented. Fig. 2.7 illustrates the delay at the output of the inverter for a different number of stacked devices. It is confirmed from Table 2.3 and Fig. 2.7 that the delay time increases as more elements are included in the array.

## 2.6. Circuit Simulation Results Summary

As we have commented, the use of stack devices increases the number of the transistors sizing options, which facilitates modern analog designs. The use of stack devices also represents some advantages, such as reduction of leakage current (stack effect) and an increase of the output

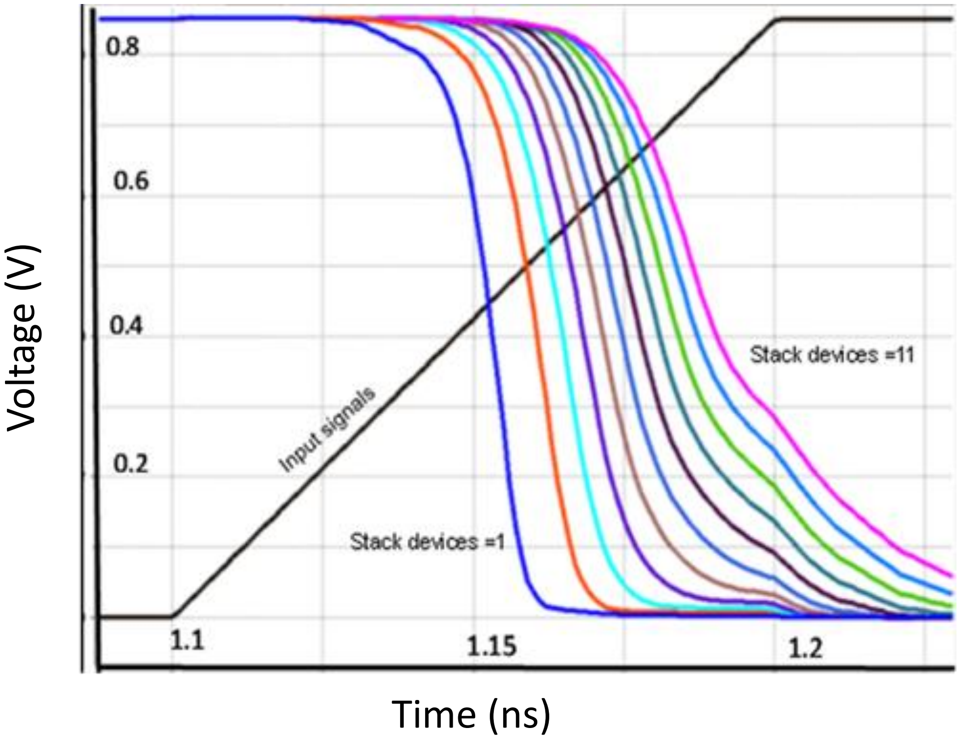


Fig. 2.7 Curves that illustrate the propagation time for a different number of stack devices in an inverter

resistance with respect to a single device. These benefits apply to standalone devices and more complex cells, such as a current mirror. However, from Table 2.3 and Table 2.4 we can notice that as the number of stacked devices increases, the improvement in the previous parameters is less significant. This saturation effect is illustrated in Fig. 2.8, where the values of the leakage current, the output resistance for standalone transistor and for a current mirror, are represented in percentage with respect to the value when a single device is used. In contrast, the delay time (one of the negatives effect of using stack devices) increases almost linearly as the number of devices of the array increases, as seen in Fig. 2.8

## 2.7. Summary

In this chapter, some of the implications of the use of the stacked transistors topology were analyzed. The use of stacked transistors increases the number of options for the design of modern analog circuits. It also offers additional advantages, such as a significant reduction of leakage

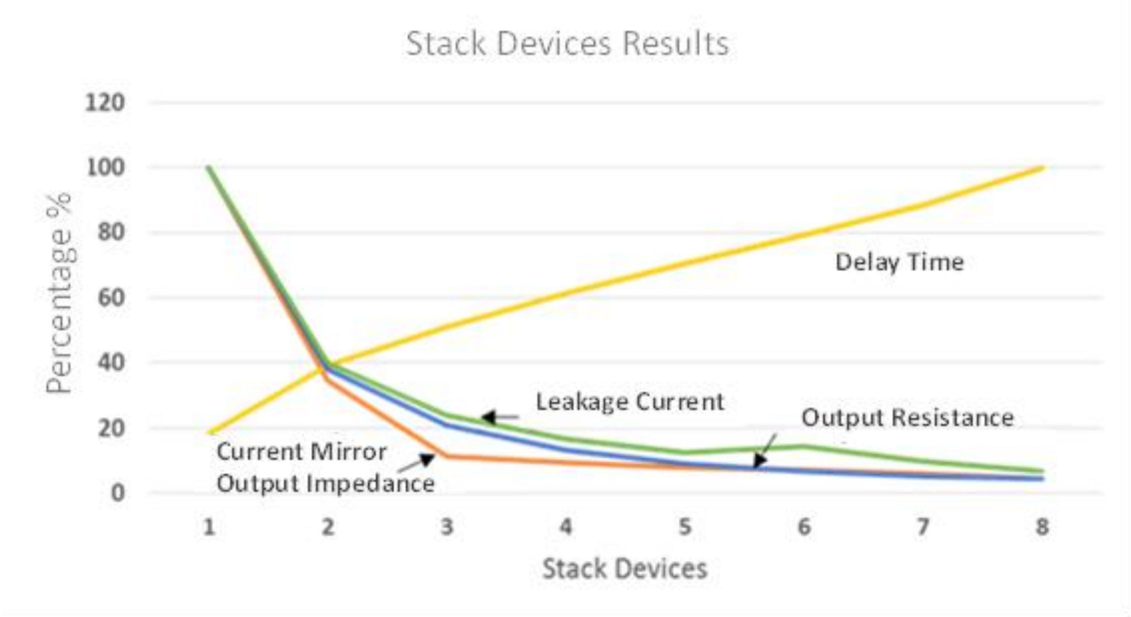


Fig. 2.8 Comparison of the reduction for the leakage current, the output resistance for a standalone transistor and for a current mirror and the increase of the delay time for a different number stacked devices.

current (stack effect) and an increase of the output resistance with respect to a single device, which is especially useful for more complex structures, as it was illustrated with the use of current mirrors. However, they also have some drawbacks, such as increased propagation delays. Also, the improvement achieved on the output resistance and leakage current is less significant as the number of the elements in the array increases. In terms of layout topologies, as more elements are added in the array, the values of the parasitic elements increase, reducing the maximum operating frequency of the circuit; when more elements are included, the required area for the layout and the complexity of its implementation increase. We can also notice that as the number of stacked devices increases, the improvement of some of the characteristics of the circuit is less significant, but the required area increases almost linearly.

The analyses presented show the importance and efficacy of these structures, as an option for the implementation of analog circuits, for facing the sizing limitations of current nanoscale technologies. This analysis of the array structures and the analysis of the different options for its layouts, presented in chapter 1 addressed the importance to have a tool that allows fast implementations of this layout options, for their efficient characterization and the later use in more complex analog circuits. In the next chapter, the description of a synthesize CAD tool, intended to accelerate the layout creation of these structures and well as the differential pair is presented.

# 3. Synthesis Tool for Automatic Layout Generation

In the previous chapters, the restrictions for the implementation of analog circuits in modern nanoscale IC fabrication technologies are addressed. Also, in chapter 1 the analyses of the layout implementation of the differential pair and stacked transistors are described. We concluded that even for these basic structures there are a large number of options to implement their layout given the large number of placement and interconnection options available, limited among other variables by the area allocated for the layout. For the implementation of the most suitable layout for a specific design, it is necessary to have a tool that helps to accelerate the layout implementation of analog structures, for their subsequent analysis, characterization, and optimization.

In this chapter, we present an analog layout synthesis tool that allows the automatic generation of multiple layouts views for the two circuits considered in this work. The proposed CAD tool has the following features:

a) The layout placement uses an internal database that includes different topologies for these two fundamental analog structures. Templates are used to indicate the list of elements and the order in which the devices should be drawn. Codification of these templates is based on systematic algorithms, allowing the addition of new topologies.

b) The user provides the names of the transistors that should be implemented in layout; the CAD tool validates that these devices form one of the structures defined in the tool. This verification is based on the interconnection of the devices regardless of the order of the selected devices, which offers the possibility to include new topologies in the tool.

c) For the placement process, each transistor finger is automatically drawn individually and then automatically replicated and placed to create a complete structure, instead of drawing a single large object. Each device has a unique list of parameters, and the selected topology from the database defines its location. Pcells are not used for the creation of the layout arrays. The user can configure the CAD tool to obtain different levels of detail from each layout; this allows the user to define which metal layers are included in the layout. Each layer, including the contacts, is also drawn individually, based on pattern templates and design rule files. These levels of detail provide the user with relevant information for analysis and comparisons of different circuits' performance.

d) All the automatically implemented layouts are short-circuit free and DRC clean, according to the selected technology process; this means that no re-routing is needed. Depending on the level of detail, the resultant layout can also be open-circuit clean and density rules compliant.

e) Due to the algorithms modularity, our tool can be migrated to other technology processes, adjusting the set of design rules (captured manually) and technology process file, allowing the possibility to work with different types of transistors.

f) The scripts that form the proposed tool are developed using TCL language. This tool 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.

Also, different tests of the proposed CAD tool are presented in this chapter. The synthesis tool is tested by generating multiple layout versions of the two structures under study, considering multiple combinations for their implementations. The parasitic capacitance and resistance are obtained for each view using a proprietary Intel Corporation extraction tool, the information provided by this process allows to understand the impact of the non-idealities on the layout and how these non-idealities affect the performance of the circuit. The scripting language used for the development of the scripts is TCL, and the technology used is a nanoscale Intel Corporation process.

## 3.1. General Tool Description

The developed CAD tool is a collection of scripts intended to speed up the layout implementation of a specific circuit structure. The general flow diagram of the CAD tool is presented in Fig. 3.1. Three main stages are considered: environment setting, placement implementation, and layout completion. The main inputs for the environment setting stage are the netlist and the names of the transistors that form the structure to be implemented in the layout. The netlist is provided as a text file, and the names of the transistors are captured manually. The CAD tool validates that these inputs form a valid structure. Also, the user can define other parameters and specifications to get different layouts of the structure; these parameters include, level of detail (selecting the metal layers to be added); the number of side and middle transistor dummies, number of rows and the insertion of substrate connections and contacts. Parameters specified on the netlist, such as transistor type and model, its width and length and its terminal connections, are directly read and saved.

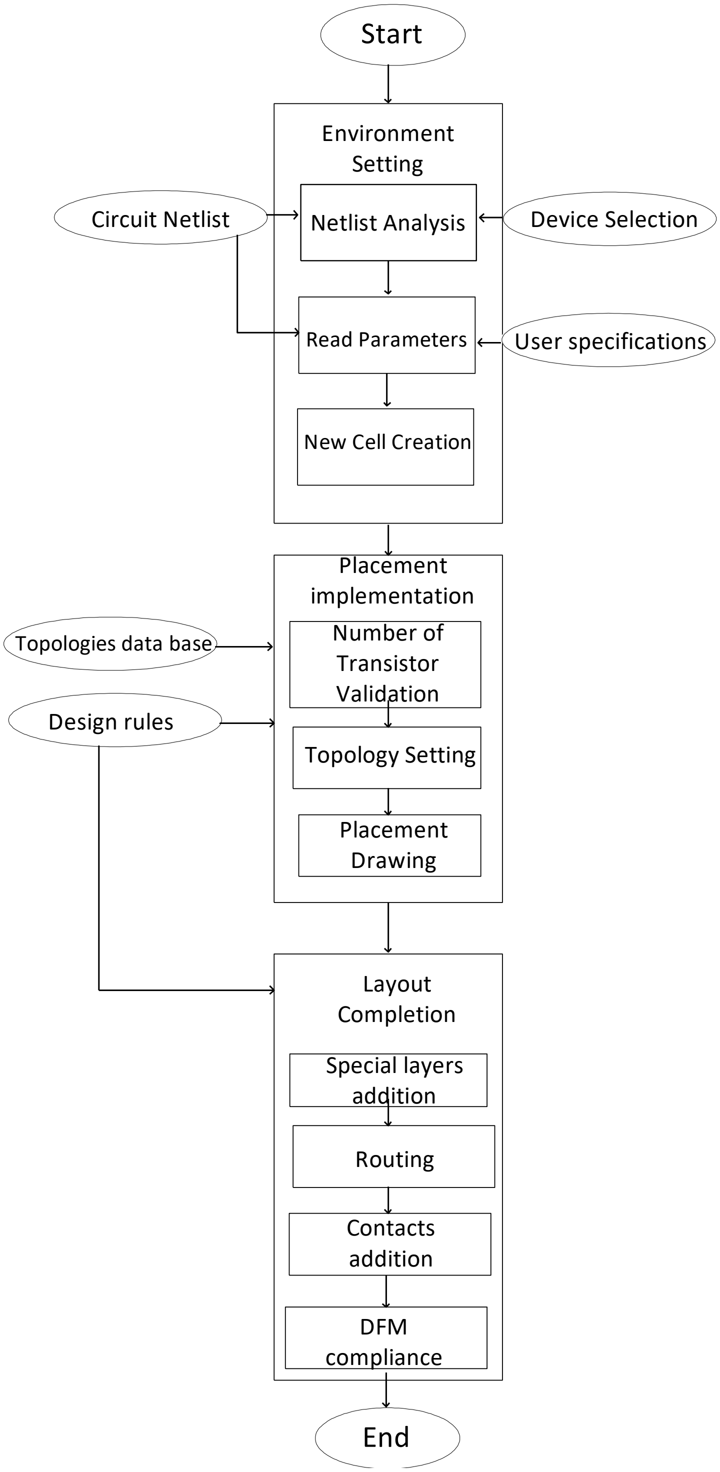


Fig. 3.1 Flow diagram of the analog layout synthesis tool.

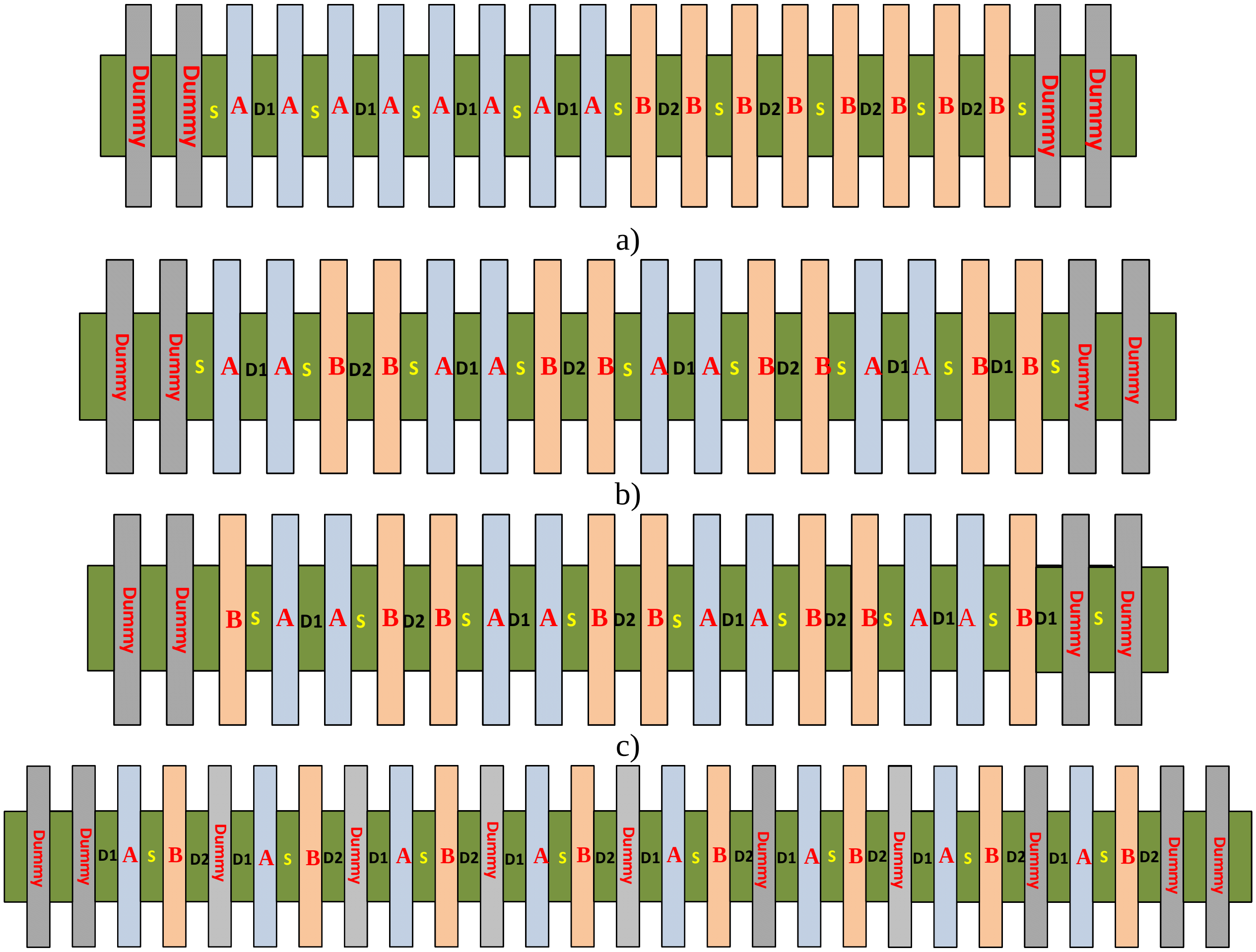
Finally, the environment setting stage creates two new cells: one for the schematic view and one for the layout view. After the creation of these new cells, the next step consists of implementing the layout, which starts with the placement implementation stage (see Fig. 3.1). The tool validates the viability of the layout implementation, according to the number of devices and rows defined. For the generation of the layout, the tool uses a database that includes well-known layout topologies for the two fundamental structures defined. The tool also includes the technology or process design rules.

In the placement implementation stage, devices are drawn individually to form the required structure. Only P or N diffusions and polysilicon layers are included, as well as the terminal names. Each transistor is drawn complying DRC rules. The substrate connections are also added in this stage.

Finally, in the layout completion stage, special layers, metal routes and their respective contacts are added. Metals added in this stage depend on the level of detail defined by the user and may include: base metal layers, metallization for interconnecting, and/or their corresponding contacts. Also, metal fill layers can be added to accomplish the Design For Manufacturing (DFM) requirements for DRC and density rules.

## 3.2. Topology Libraries

Similarly to the approach presented in [Chávez-Hurtado-09], the proposed CAD tool contains a database that includes different topologies for the two fundamental structures considered: stacked devices and differential pair. The CAD tool is prepared for future additions of other fundamental structures. The templates of the different topologies considered in the CAD tool are saved and implemented as libraries. In section 1.8, different options for the implementation of these two structures are described. From these options, four different implementations are chosen for the differential pair templates: the One Share Diffusion, the Interdigitated Layout, the Common Centroid, and the Interdigitated Layout with dummy devices. For the library of the stacked devices, three templates are defined: the One Share Diffusion, the Interdigitated Layout, and the Interdigitated Layout with dummy devices. All these implementations are briefly described as follows.


Dummy Dummy S A D1 A S A D1 A S A D1 A S A D1 A S B D2 B S B D2 B S B D2 B S B D2 B S Dummy Dummy
Dummy Dummy S A D1 A S B D2 B S A D1 A S B D2 B S A D1 A S B D2 B S A D1 A S B D1 B S Dummy Dummy
Dummy Dummy B S A D1 A S B D2 B S A D1 A S B D2 B S A D1 A S B D2 B S A D1 A S B D1 Dummy S Dummy
Dummy Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy D1 A S B D2 Dummy Dummy

a)

b)

c)

d)

Fig. 3.2 Differential pair layout topologies: a) One Shared Diffusion; b) Interdigitated Layout; c) Common Centroid; d) Interdigitated with dummies.

### 3.2.1 Differential Pair Topologies

The layout topologies currently implemented for the differential pair are shown in Fig. 3.2a-d.

a) One Shared diffusion (see Fig. 3.2a). In this topology the transistors are placed next to each other: all the fingers of one of the transistors are placed next to all the fingers of the second transistor.

b) Interdigitated Layout implementation (see Fig. 3.2b). The fingers of the transistors are placed alternating two fingers of the first one and then two fingers of the second one.

c) Common Centroid Layout (see Fig. 3.2). As in the interdigitated layout implementation, the fingers of the transistors are placed alternated in pairs, but the distribution is modified to achieve a gradient tolerance on X axes.

d) Interdigitated layout with dummy insertion (see Fig. 3.2d). Between each pair of active transistor fingers, a couple of dummy devices are added, which reduces the diffusion area of the source and allows having an odd number of fingers.

### 3.2.2 Stacked Devices Topologies

The layout topologies for the stacked devices are shown in Fig. 3.3, using as an example, an array of four stacked devices with a number of fingers per transistor equal to three. The layout topologies included in the library of the proposed CAD tool are the following:

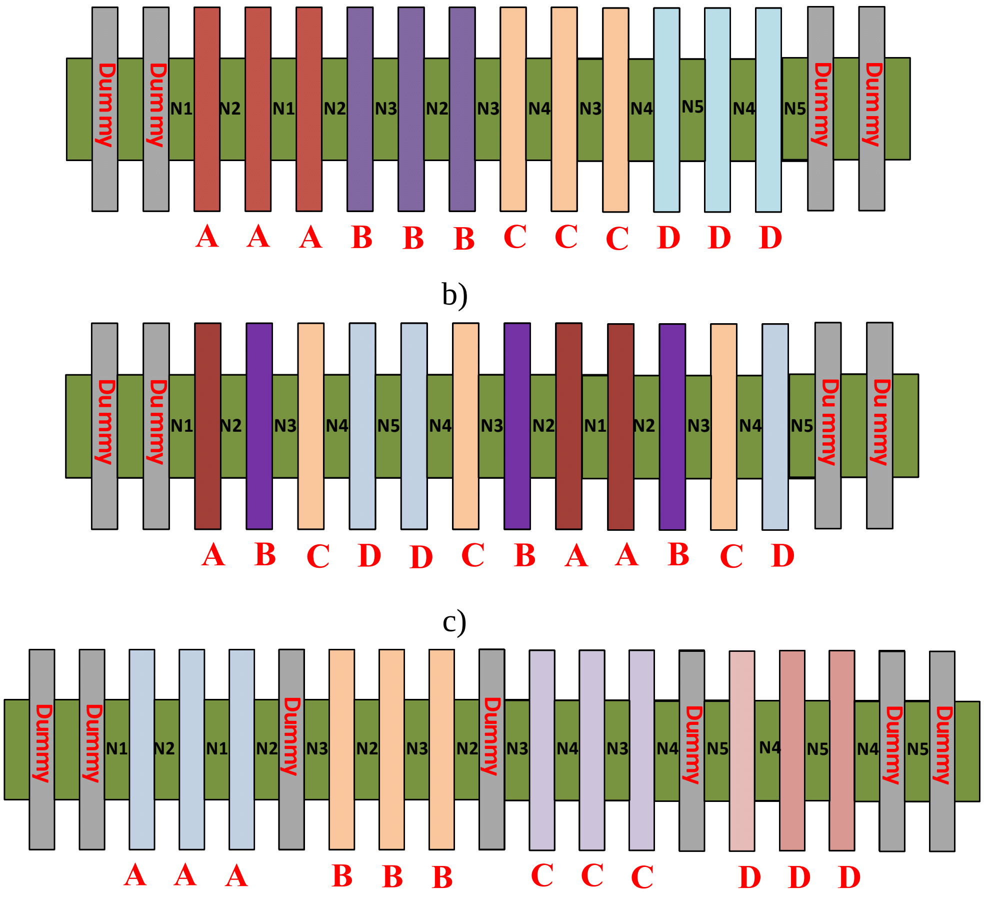
a) One Shared Diffusion (Fig. 3.3a). Similarly to the case differential pair topology, all the fingers of one of the transistors are placed next to each other; for this, the number of fingers of each transistor must be an odd number.

b) Interdigitated Layout implementation (Fig. 3.3b). Only one finger of each transistor is placed next to each other, starting from the transistor in the "top" of the array and continuing until one finger of the transistor that is on the "bottom" is placed; then the order is inverted, from the bottom to top. The process is repeated until all the transistors fingers are placed.

c) Interdigitated with dummy insertion (Fig. 3.3c). As in the case of the interdigitated layout dummy devices can be added between the fingers of the transistors; this allows the use of an even number of fingers; also helps to reduce the self-heating effect.

## 3.3. Environment Setting

In this stage (from the flow of the synthesis tool presented in Fig. 3.1), the user-defined specifications and parameters for the implementation of the layout are validated and saved. Parameters defined on the netlist are also obtained and saved. Two new cells are created: one for the schematic view and another one for the layout generation.



b)

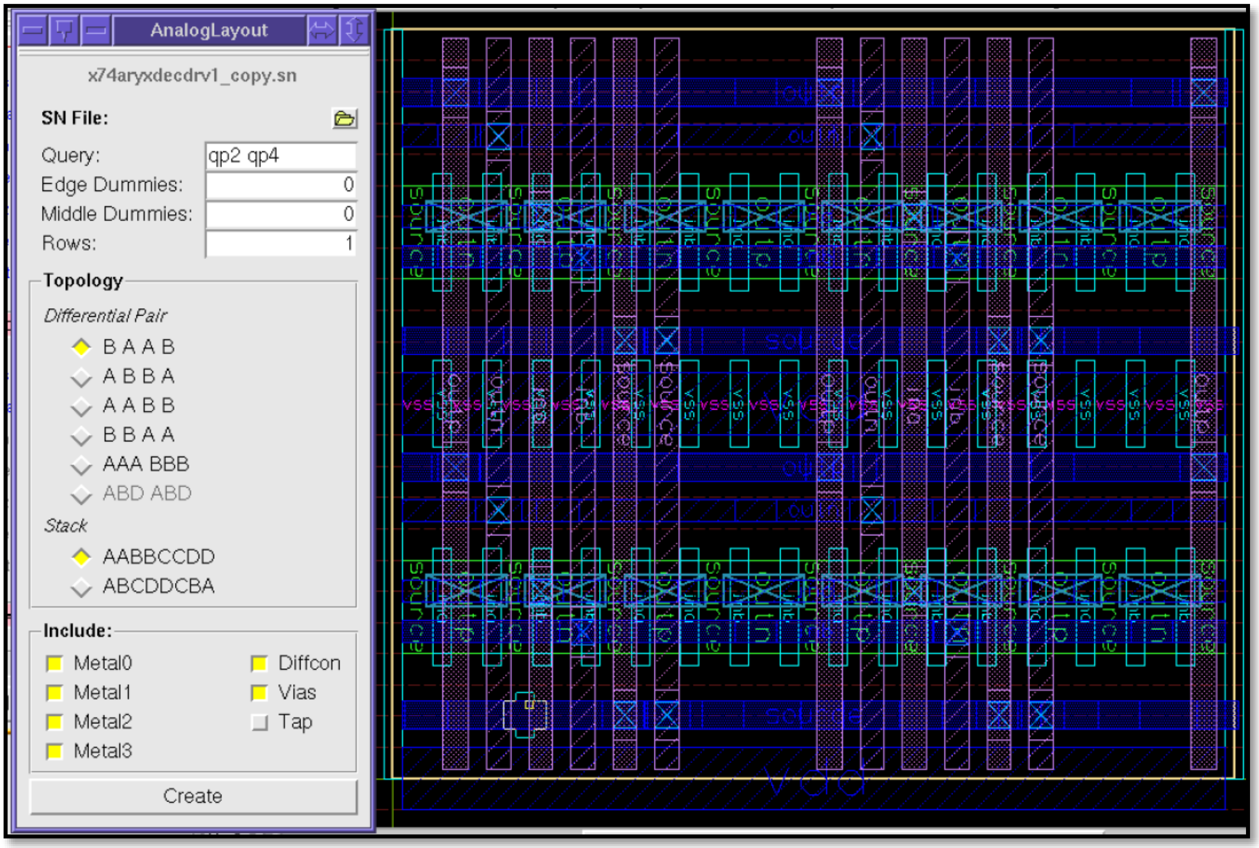
c)

d)

Fig. 3.3 Layout topologies for an array of four stacked devices with a NFPT =3: a) one shared diffusion; b) interdigitated layout; c) interdigitated with dummy insertion.

### 3.3.1 Main Window

A graphics user interface (GUI) was created to facilitate our CAD tool utilization. Through this GUI the user can select the netlist file of the circuit where the structure to be implemented in the layout is defined. The user can type the names of the transistors that are used in the implementation of the desired structures. Through selection boxes, the user can define the required topology for the layout and its level of detail. In addition, the user can optionally add the corresponding contacts and the substrate connection. Other options that can be selected are the number of dummy devices at the end and in the middle of the array, as well as the number of rows in which the layout is implemented. In Fig. 3.4 is shown the GUI of the proposed tool and a complete layout generated by it.



AnalogLayout

x74aryxdecdrv1_copy.sn

SN File:

Query: qp2 qp4

Edge Dummies: 0

Middle Dummies: 0

Rows: 1

Topology

*Differential Pair*

- ◆ B A A B
- ◇ A B B A
- ◇ A A B B
- ◇ B B A A
- ◇ AAA BBB
- ◇ ABD ABD

*Stack*

- ◆ AABBCCDD
- ◇ ABCDDCBA

Include:

- ☑ Metal0
- ☑ Metal1
- ☑ Metal2
- ☑ Metal3
- ☑ Diffcon
- ☑ Vias
- □ Tap

Create

Fig. 3.4 Image of the proposed CAD tool and a generated layout example.

### 3.3.2 Netlist Analysis

A dedicated function reads the netlist and analyses the connections of the selected transistors, indicating what kind of structure these devices are building: a differential pair, an array of stacked devices, or neither of them. The Netlist Analysis follows the next steps:

a) The function receives as inputs the name of the netlist file and a list of transistor’s names. Since the netlist is provided from the GUI, it is guaranteed that the file already exists.

b) An iterative search parses all the devices related to the desired selection to verify that all instances of each transistor have the same connections or terminals; this means that all the device's instances are connected in parallel. When there is an instance that is not in parallel, an error is flagged.

c) To analyze the connections between devices and verify that a valid structure is formed, two verification processes are used:

i. Verification for the differential pair: This process is only used when the number of selected devices is two; checking that the source connections are the same for both devices while the drain and gate ones should be different.

ii. Verification for the stacked devices: This process is used when the verification for the differential pair fails or when the number of selected devices is larger than two. The process verifies that the gate connection is the same for all the devices. It also verifies that at least one of the other two terminals (source or drain) of each transistor is shared with the opposite of another transistor (e.g., the source connection of one device is shared with the drain connection of any other transistor), checking that there are only one source and one drain connection per net in the design. Fulfilling these conditions guarantees that the selected transistors form an array of stacked devices. If this second verification process fails, an error message is flagged.

## 3.4. Read Parameters

When a valid structure has been identified, the next step is to get the rest of the transistors parameters from the netlist, which are the transistor's width, length, model, and type. These parameters and the terminal connections are saved in a register to be used as inputs for the next processes. Additionally, the parameters defined by the user through the GUI are saved on specific registers.

### 3.4.1 Cell Creation

In this stage, two new files are created: one for a new schematic view and another one for the layout view. The new schematic view includes only the devices and their corresponding connections that were selected in the GUI. The name of the file is unique and is formed by a reference of the structure to be implemented (diff_pair or stack) and an index number. The name of the file for the layout is the same name used for the schematic view, except for the extension. The next stage is the creation of the layout itself, starting with the placement of the devices.

## 3.5. Placement Implementation

The process for the layout placement implementation is divided into three basic functions namely: Number of transistors validation, Topology setting, and Placement drawing. Before the description of these functions, some considerations are needed for the generation of the layout, which are described next.

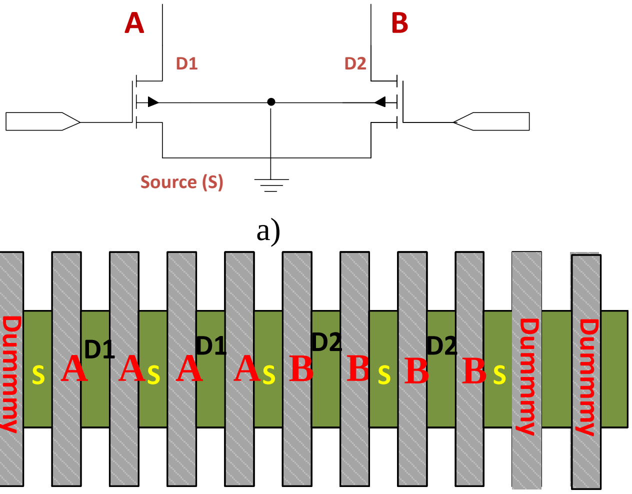
### 3.5.1 Layout Considerations

As it was mentioned before, we use an Intel Corporation fabrication process for the implementation of the layouts. The characteristics of the layouts are determined by the constraints of the process, some of those constraints include:

a) The use of discrete values for transistor's width: their dimensions are multiples of a minimum value ($W_{\text{min}}$). However, in this proposal, the values used for the transistor width are dimensions that enable the placement of an even number of contacts (or vias). The widths used for the transistors implementations are two, four, and six times $W_{\text{min}}$.
b) Use of guides or tracks for metal/layer routing.
c) All the transistors are vertically oriented.
d) All the metals have a specific direction: M0, M2 are drawn horizontally, while M1 and M3 are drawn vertically.

### 3.5.2 Number of Transistors Validation

Here we validate that the layout can be implemented based on the number of rows and the number of fingers of each transistor; the total sum of transistors is given by the number of fingers of each device and the number of dummy devices at the ends and in the middle of the array. This number should be divisible by the number of selected rows. If it is not possible to create the layout using the desired number of rows an error is flagged, asking the user to make a new selection. For instance, let us consider an array of three stacked devices, each of them with three fingers. This layout cannot be implemented in two rows since nine devices cannot be placed evenly in two rows.



a)

b)

Fig. 3.5 Example of placement creation: a) differential pair formed by transistors A and B, each of them with two fingers; b) layout implementation on one row adding two dummy devices and using one source shared topology.

One consideration to validate the viability of the layout implementation is that the number of dummy devices at the sides of the array, which implies that this number is repeated for each row. For example, if two side dummies are selected, and the layout is implemented in two rows, a total number of eight dummy devices are placed. There are some topologies where the number of fingers of each device should be an odd number; our tool also validates those cases.

### 3.5.3 Topology Setting

Once the total number of transistors has been validated, the next step is to define the order in which they are laid out. This order depends on the structure (differential pair or stacked devices), the selected topology, the number of fingers, the number of dummy devices, and the number of rows. The list of devices is entered in a register, in the order in which they should be placed in the layout, starting from the bottom left and ended on the top right. For instance, let us consider the circuit of Fig. 3.5a: a differential pair formed by transistors A and B, each of them with four fingers. Consider that two dummy devices are desired at each side of the layout array, that the layout is implemented in one row, and the topology to be used is One Shared Diffusion (see Section 3.1). The list with the placement order for the devices is: [D D A A A A B B B B D D] where A indicates the device A, B the device B, and D the dummy devices. The corresponding layout is shown in Fig. 3.5b. Once the order of the devices has been set, the tool proceeds to do the placement.

### 3.5.4 Placement Drawing

To draw the layout placement, each device is created individually, following the order defined in the topology setting function and according to the parameters obtained from the netlist. To do this, a sub-function named “Transistor Generator” is employed.

### 3.5.5 Transistor Generator

Transistor Generator is the essential sub-function included in the CAD tool. It first cleans up any possible element (layer, device, etc.) that could exist in the layout cell where the layout is created. Next, it generates every single transistor by drawing it on a specific location of the cell that was previously created. The transistor drawing includes only the diffusion and polysilicon layers. For the drawing of each device, it is necessary to provide the following parameters:

a) Type of transistor (NMOS or PMOS).

b) Transistor model (high speed, low current leakage, low power consumption, etc.).

c) Transistor width and length.

d) Terminal connections: net names at which the transistor terminals are connected. It is necessary to specify individually, the names of the drain, gate, and source and bulk terminals.

e) Transformation: parameter to indicate if it is needed to flip the transistors (to interchange the order of the device’s terminals).

f) Rotation. In most of the current process technologies, only one direction is allowed to draw the transistors. Typically, the transistor gate should be perpendicular to the "X" axis; if required, this parameter indicates how many degrees the transistor is rotated with respect to the "Y" axis; by default, its value is 0.

g) Translations on X and Y; these parameters specify the exact locations where the transistor should be drawn with respect to the origin point of the cell. These are illustrated in Fig. 3.6.

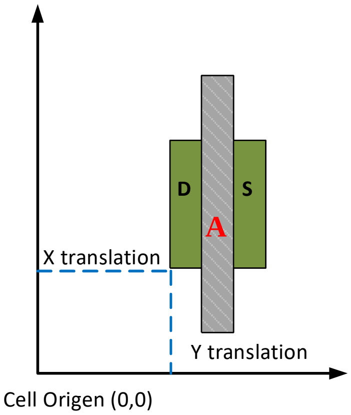


Fig. 3.6 Translations on X and Y axis: these parameters specify the exact location where the transistor is drawn.

With this information, it is possible to draw a complete transistor with all the required parameters or specifications, on an exact location of the new cell. Since each device to be implemented in the layout has a unique position, it is necessary to have a control function that indicates to the transistor generator where to draw each of them. A placement drawing control sub-function makes a sweep through all the devices to be drawn in the cell and defines the transformation, rotation, and translations parameters required.

### 3.5.6 Placement Drawing Control

The placement drawing control sub-function is shown in Fig. 3.7. Two loops form it. The first one is used to count the number of rows (row_count) in which the layout is implemented, while the second one is used to count the transistors (tran_count) that are placed on each of the rows. The layout is drawn transistor by transistor and row by row, following the order of the list defined by the topology setting function. Transistors are drawn starting from the bottom-left and finishing at the top-right. A pointer (next_tran) is used for the register created on the topology setting function (device_list) to get the next device to be drawn; once this device is obtained and based on which kind of device is (dummy, active or tap device), a process call Terminal Setting determines the terminal connections and if a rotation or transformation is needed on the device.

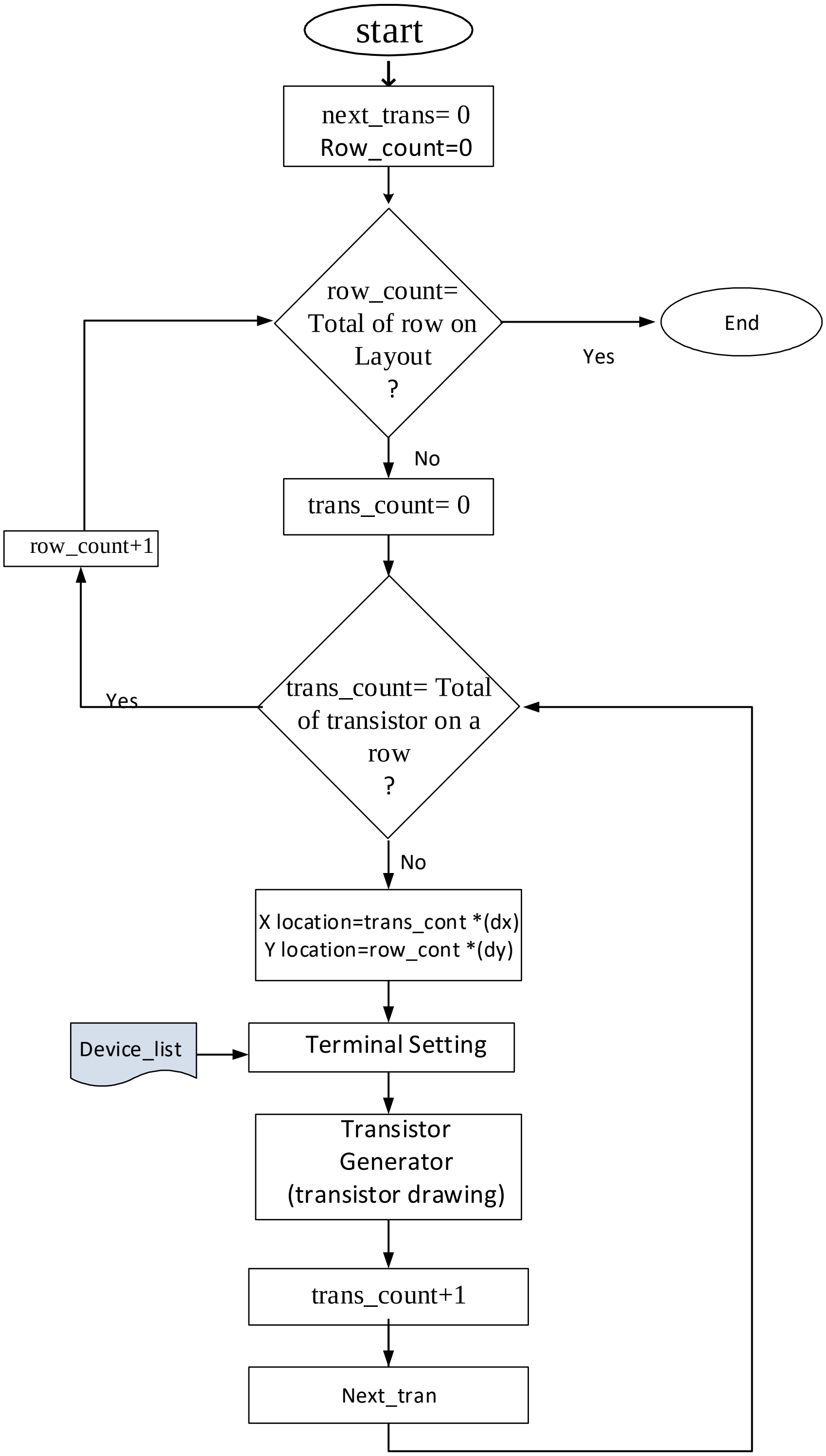


Fig. 3.7 Flow diagram of the placement drawing algorithm.

Then it is necessary to define where the transistors are drawn. In most of the technologies transistors are placed uniformly on the X-axis of the cell, this means that all the transistors are placed using a discrete grid, and their separation is uniform.

The same situation occurs for the layout rows, which are distributed evenly along the Y-axis. Based on this, two variables are created, Δx and Δy, to define the transistor location. The specific position in which the transistor is placed is determined by the product of Δx and trans_count variables for X coordinate (with X translation parameter) and by the product of Δy and row_count variables, for Y coordinate (with Y translation parameter).

Once the above parameters are defined, the Transistor Generator function draws the device in the corresponding location. Once the device is drawn, tran_count and next_tran are incremented to get the next device to be implemented in the layout. Once all the devices of the current row are created and drawn in layout, tran_count is set to zero, while row_count is incremented to continue drawing. When all the transistors from all the rows have been placed, the placement procedure is completed.

## 3.6. Layout Completion

The last function of the proposed CAD tool is the layout completion, where the remaining layout layers, metal routes and contacts are added. Through the GUI, users can select the level of detail required for their layout: placement of the devices, basic metals, metallization to interconnects, or DFM compliance. The user can also select if the corresponding contacts should be added. A description of these levels of detail follows.

a) Placement of the devices. So far the layout is formed only by the diffusion layers (diffusion N or P), and the polysilicon (poly) wires. Here, the N-well (for PMOS devices), special ID layers, and contacts can be optionally included.

b) Basic metals. The connection between different devices or transistors is made through different metal layers. Base metal layers are not generally used for long interconnections due to their high resistance. Their function is to create a path from the base layers, polysilicon, and diffusion, to the first metal used for interconnection. These layers have fixed width values defined by the design rules, and their location is also fixed to specify layer grids. Due to these restrictions, the possible variations of the basic metal routing depend mostly on the selected topology. At this level of implementation, there is not a “complete connection” between the devices, even if the contacts are added.

c) Metallization for interconnects. Metal layers used for routing or interconnection are the interconnection layers. These layers offer multiple options for their width value. The spacing rules between the different metal tracks depend on the width selected for them. All these possible combinations of width and space provide a large number of options for signal interconnections. The CAD tool uses a standard metal pattern, where all the metal tracks, excepting those for power, have the same width and separation. Finally, if contacts are added, the layout implementation is LVS clean, and a complete connection between all devices is achieved.

DFM compliance. Fill layers should be added to accomplish DFM requirements, including DRC and density rules. The resultant layout version accomplishes all the verification checks, such that it can be incorporated into more complex layout designs as an “analog standard cell." The creation of multiple versions, with this detail level, may allow the designers the creation of their analog layout library.

## 3.7. Analysis of the Metallization on Layout Implementations

The proposed analog layout synthesis tool is tested by generating multiple layout versions for the differential pair and stacked devices structures. The configurations and parameters are defined individually for each layout through the GUI. The primary intention of this test is to assess the quality of the layouts and their time of implementation. The resultant layout implementations are DRC clean for width, spacing and enclosure rules [Lomelí-Illescas-16]; all the implementations are short-circuit free (with no short-circuit errors; no manual fix required); this is verified through a layout versus schematic (LVS) flow [Lomelí-Illescas-16]. For those cases where the level of detail includes metallization, the layouts were also open-circuit free; the execution time consumed for the generation of each of these layouts was less than a minute. These results show the efficiency of the proposed tool for the layout implementation of basic structures. To compare the different implemented layouts and to illustrate the kind of information that can be obtained from them, a parasitic extraction process was executed over each layout. The algorithm used to perform the parasitic extraction is available in an internal design tool of Intel Corporation. This algorithm is executed, manually by the proposed CAD tool, over each layout after its creation. The information relative to the parasitic elements (resistance and capacitance only) is obtained from the report files that the extraction algorithm generates. Some tests and analyses were performed to compare the effects of different layout implementations; two test cases for the structures considered in this thesis are presented in the next subsections.

### 3.7.1 Differential Pair Study Case 1

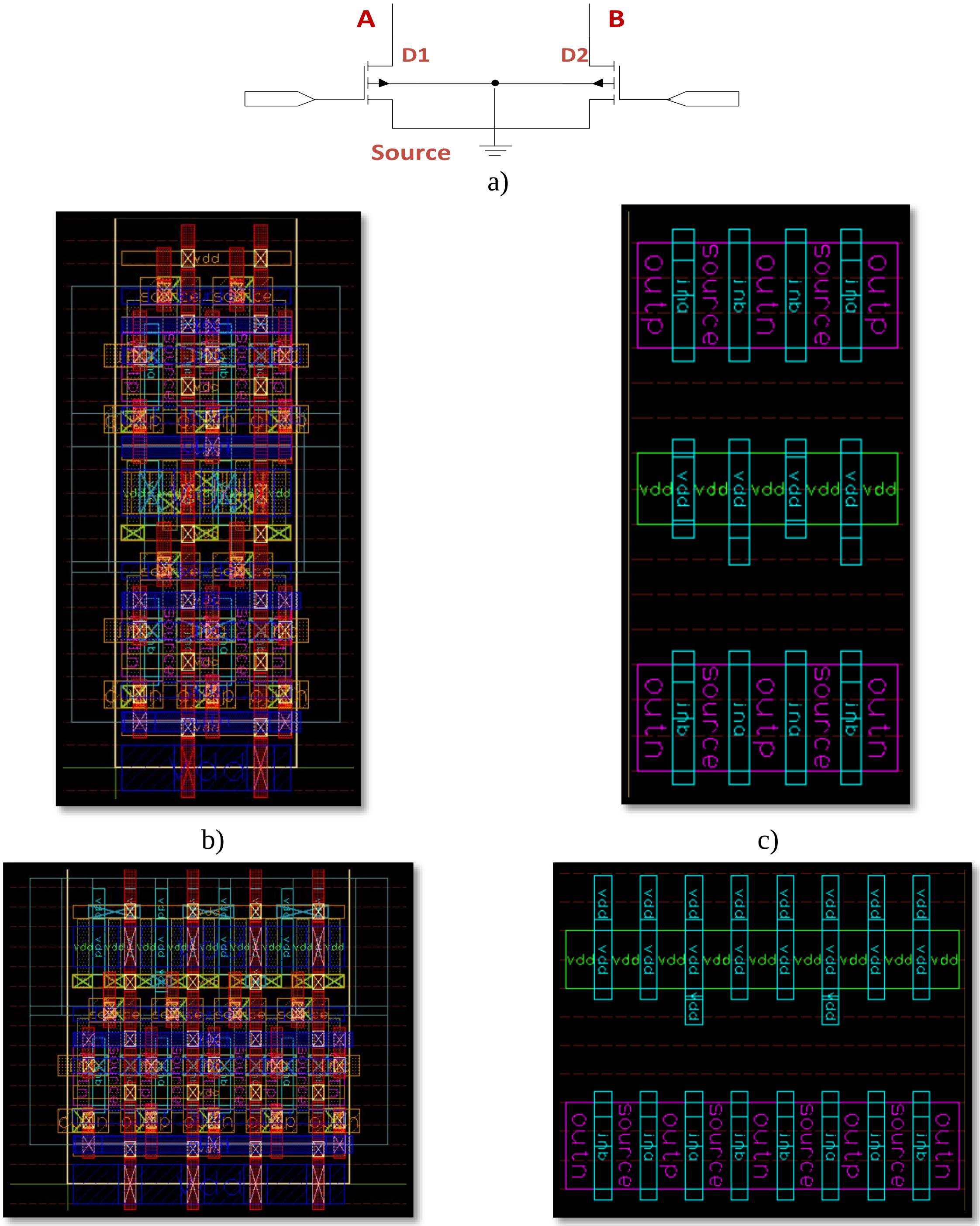
In this case, three different topologies for the differential pair structure are compared: One Shared Diffusion, Interdigitated Layout, and Common Centroid layout, which are described in section 3.2.1. A differential pair formed by PMOS transistors A and B, each of them with 8 fingers, is shown in Fig. 3.8a. This circuit is laid out using the three possible topologies and two different levels of detail: the first one including the metallization for interconnecting and the second one considering only the placement; the substrate connections were added in all the cases; no dummy devices were added, and each layout was implemented in one or two rows.

To compare the layouts, the capacitance from power net ($V_{DD}$) to the rest of the layout elements, (cross capacitance, $C_C$) and its connections resistance ($R_C$) were extracted using the internal extraction tool. The results are summarized in Table 3.1. From that table, we can see how the different layout implementations can affect the capacitance and resistance between the different elements/nets of the circuit. We can observe that if we consider only the placement of the devices, each topology by itself does not have a significant effect on the parasitic component values; however, when the rest of layout elements (metal layers and contacts) are added, the values for the parasitic elements change from one topology to another. The selected topology determines the location of the devices/transistors and also how the connections between these elements is implemented; this means that the chosen topology determines the length and location of the metal used for interconnection and their corresponding contacts; these differences have an effect on the values of the capacitance and resistance elements, as observed in Table 3.1. Similarly, when only the placement of the devices is considered, the values of cross capacitance and resistance are very similar even if the layout is implemented on one row or is implemented using two rows; this is because when only the placement is considered, only P diffusion and polysilicon layers are involved in the generation of parasitic elements. Since the number of devices is the same for

TABLE 3.1. SUMMARY OF LAYOUT IMPLEMENTATIONS OF A PMOS DIFFERENTIAL PAIR

| Topology | Metallization for interconnection | | | Topology | Including only placement | | |
|---|---|---|---|---|---|---|---|
| | #Rows | Rc (Ω) | $C_C$ (pf) | | #Rows | Rc (Ω) | $C_C$ (pf) |
| Common centroid | 1 | 14190 | 1290 | Common centroid | 1 | 7500 | 199 |
| Common interdigitation | 1 | 13700 | 1270 | Common interdigitation | 1 | 7503 | 198 |
| Devices next to each other | 1 | 13420 | 17400 | Devices next to each other | 1 | 7510 | 197 |
| Common centroid | 2 | 6100 | 887 | Common centroid | 2 | 3930 | 228 |
| Common interdigitation | 2 | 5589 | 929 | Common interdigitation | 2 | 3920 | 225 |
| Devices next to each other | 2 | 7060 | 1020 | Devices next to each other | 2 | 3980 | 234 |

implementations in one row or two rows, the value of the parasitic elements is very similar. When metal layers and contacts are added to the layout, the number of parasitic elements increases due to the multiple variations in the routing of the nets. If the layout is implemented using two rows instead of one row, the number of metal tracks for interconnections and the number of contacts are larger, which causes the difference of the cross capacitance and resistance values shown in Table 2.1. In Fig. 3.8 are shown the layout implementations of the common centroid layout topology, using the two different levels of details; in Fig. 3.8b and Fig. 3.8c the implementations on one row are depicted, including the metallization for interconnecting and considering only the placement, respectively. In Fig. 3.8d and Fig. 3.8e the implementations using the two levels of detail, but implemented on two rows are illustrated. As commented before, all layout implementations are DRC clean and short-circuit clean; when the complete metallization is included, the layouts are also open-circuit clean.



a)

b)

c)

d)

e)

Fig. 3.8 Layout implementation for differential pair circuit formed by PMOS transistors A and B, each of them with 8 fingers using a centroid common topology: a) schematic diagram; b) implementation on one row using all the metallization for interconnection c) placement implementation on one row; d) implementation on two rows using all the metallization for interconnection e) placement implementation on two rows.

### 3.7.2 Differential Pair Study Case2

For this test case, a differential pair formed by the NMOS transistors A and B, each of them with 16 fingers is considered which is shown in Fig. 3.9a. This circuit is laid out using just two possible topologies: One Shared Diffusion and Common Centroid layout implementation, both with the same levels of detail used in the previous case, including the metallization for interconnecting and considering only the placement. Substrate connections are added in all the cases. Each layout is implemented with and without a couple of dummies at the side of the array and using one or two rows. In this test, the cross capacitance of the nets D1, G1 and S are extracted. These nets correspond to the drain, gate, and source terminals of the transistor A. Normally, the drain and gate nets correspond to the output and input terminals, respectively, of an operational amplifier. The extraction of this value allows us to compare the possible layouts effects over the input and output load on an operational amplifier circuit.

The corresponding results are summarized in Table 3.2. From this table, we can see that, as in the previous case, different layout implementations affect the capacitance between the different elements/nets of the circuit. As occurred in the previous test case, the variation on the placement by itself has less significant effects on the values of the parasitic capacitance, than when the rest of the metals are included. As it was commented before, when only the placement is considered, the parasitic elements are generated by the interaction of the diffusion and polysilicon layers; since in all the configurations the number of transistors is the same, the diffusion and polysilicon areas are very similar; in consequence, the values of parasitic elements are very similar too, and their variations are caused only by their location, which is defined by the topology. When the rest of the metals are included, the variations on parasitic capacitance are more significant since more elements interacting between them. Also, as it was previously mentioned, the topologies define the length and the location of the metal layers used for the interconnection of the transistors; this increases the differences in the values of the parasitic elements.

From the Table 3.2, we can see an interesting phenomenon: the parasitic capacitance values are smaller for the implementation on two rows than that one for one row. The reason is that even when two rows are used, the number of elements and interconnections increase, generally in higher metal layers; in most of the technologies, the more upper metal layers are less resistive and capacitive than lower metal layers [Razavi-01]. In this particular case, for the implementation on

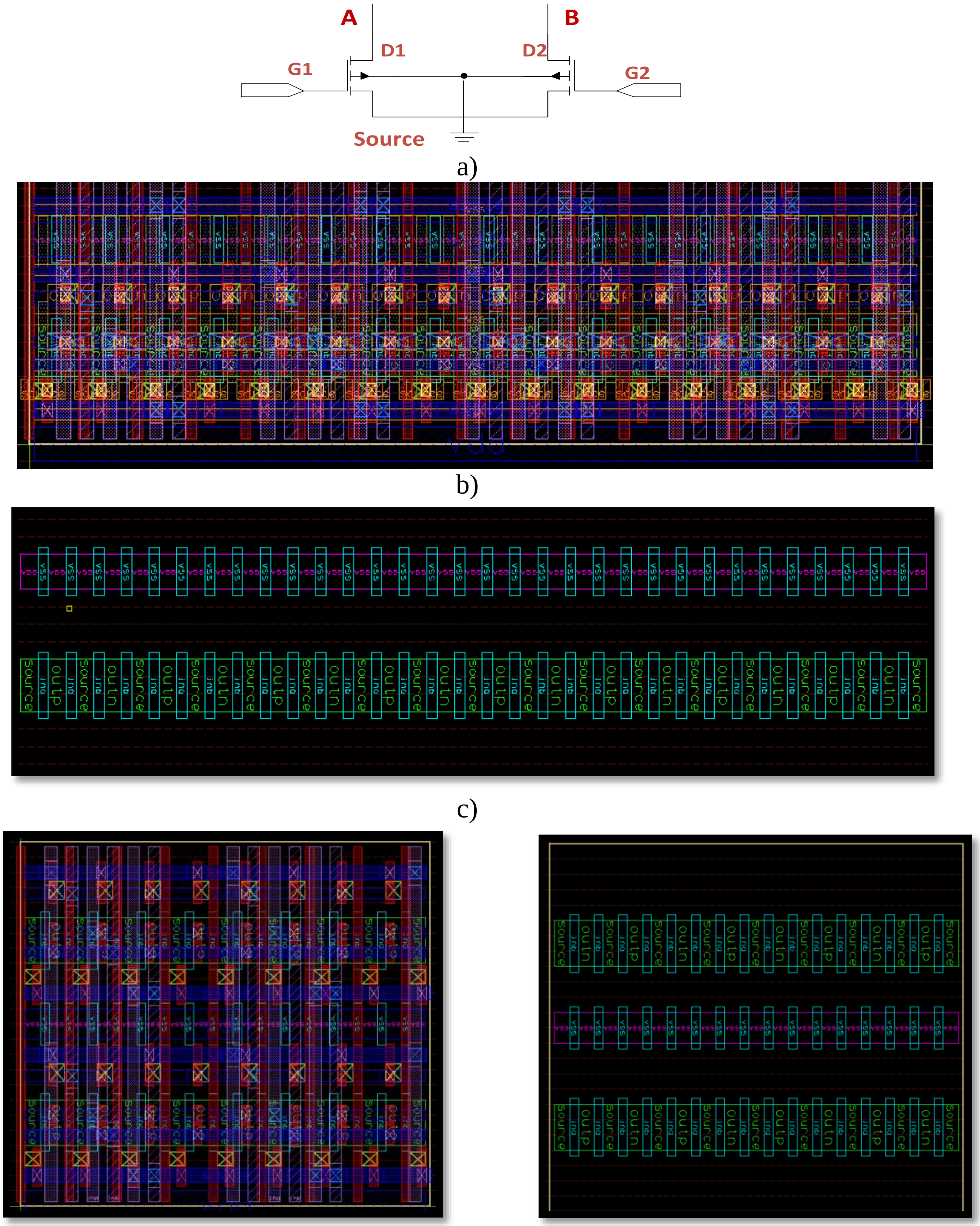
TABLE 3.2. SUMMARY OF LAYOUT IMPLEMENTATIONS OF A NMOS DIFFERENTIAL PAIR

| Topology | Metallization for interconnection | | | | | Including only placement | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Rows | Dummy Devices | $C_{g1}$ (pF) | $C_{D1}$ (pF) | $C_{source}$ (pF) | #Rows | Dummy Devices | $C_{g1}$ (pF) | $C_{D1}$ (pF) | $C_{source}$ (pF) |
| One shared diffusion | 1 | 0 | 2170 | 2230 | 3150 | 1 | 0 | 561 | 497 | 1180 |
| Common centroid | 1 | 0 | 2180 | 2460 | 2920 | 1 | 0 | 561 | 526 | 987 |
| One shared diffusion | 2 | 0 | 1650 | 1800 | 2270 | 2 | 0 | 565 | 500 | 1300 |
| Common centroid | 2 | 0 | 1610 | 1470 | 2630 | 2 | 0 | 565 | 816 | 986 |
| One shared diffusion | 1 | 2 | 2380 | 2260 | 3240 | 1 | 2 | 561 | 497 | 990 |
| Common centroid | 1 | 2 | 2380 | 2220 | 3210 | 1 | 2 | 561 | 589 | 984 |
| One shared diffusion | 2 | 2 | 1680 | 1710 | 2670 | 2 | 2 | 565 | 500 | 1070 |
| Common centroid | 2 | 2 | 1670 | 1690 | 2540 | 2 | 2 | 561 | 591 | 980 |

one row, we are using long connections on lower metals; while for the implementations on two rows the number of interconnections increased, but most of them are on higher metal layers, which reduces the value of the parasitic capacitance. These results are examples of the kind of valuable information we can get from fast synthesis layout implementations.

In the case of dummy devices, as expected, these devices have an impact over the nets they are connected, increasing the capacitance. This is a very important aspect to consider when a layout is implemented: the dummy devices are connected to the nets that are on the sides of transistor arrays; these nets could be the source or the drain of the transistor; as it was commented before, the transistor drain usually is associated with the output terminal of an operational amplifier; if just one of the outputs is connected to the dummy devices, the matching between these outputs signals may be compromised. This case illustrates the importance for the designers to understand and identify which of the topologies is the best for their designs; some topologies, as the common centroid, match the effect of the dummy devices of the output signals.

In Fig. 3.9 are shown the layout implementations of the common centroid layout topology, using the two different levels of details. Fig. 3.9b and Fig. 3.9c show the implementations on one row, including the metallization for interconnecting and considering only the placement, respectively. Fig. 3.9d and Fig. 3.9e depict the implementations using these two levels of detail but implemented on two rows. As in the previous case, all layout implementations were DRC clean and short-circuit clean; when the complete metallization is included, the layouts are also open-circuit clean.



a)

b)

c)

d)

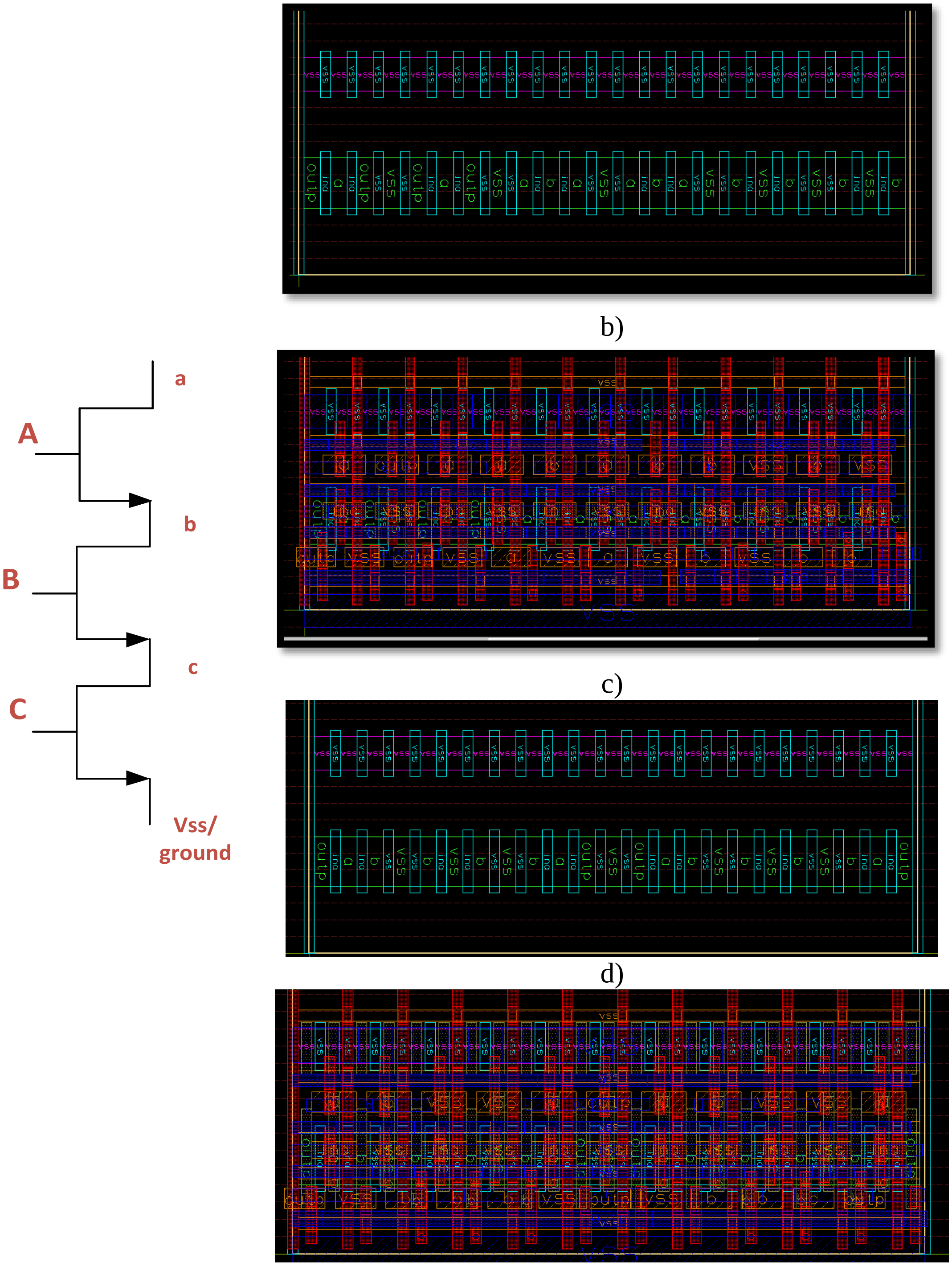
e)

Fig. 3.9 Layout implementation for differential pair circuit formed by NMOS transistors A and B, each of them with 16 fingers using a centroid common topology: a) schematic diagram; b) implementation on one row using all the metallization for interconnection; c) placement implementation on one row; d) implementation on two rows using all the metallization for interconnection d) placement implementation on two rows.

### 3.7.3 Stacked Devices Study Case1

For this test case, the circuit under consideration is shown in Fig. 3.10: an NMOS array of three stacked devices, A, B and C, each of them with four fingers. This circuit was laid out using just two possible topologies: One Shared Diffusion and Interdigitated Layout implementation, both with the same level of detail used in the previous cases (including the metallization for interconnecting and considering only the placement). Substrate connections are added in all cases. Each layout was implemented on one row, and no dummy devices were added. In this test, the cross capacitance and interconnection resistance of the nodes a, b, and c were extracted. These nets correspond to the interconnections points of the transistors; the reason to choose these nodes is to observe and analyze how the topologies and metal interconnections can affect the way the current flow through this path due to possible differences between the capacitance and resistance associated to these nodes.

The corresponding results are summarized in Table 3.3. As occurred in the previous test cases, the placement by itself has a less significant effect on the values of the parasitic elements, than when the rest of the metals are included. The intention here was to compare the values of the parasitic elements on the interconnection nodes (particularly nodes b and c) and to analyze the effects of the topologies and interconnection metals over them. From Table 3.3 we can notice that for the shared diffusion implementations, the variation on the parasitic values are smaller than for the interdigitated layout; in principle, designers should expect the interdigitated implementations have a better matching than the shared diffusion implementations, since the transistor array has smaller dispersion and better symmetry between them; these characteristics are some of the most critical factors to get a perfect match between two devices [Lomelí-Illescas-14]. In contrast, for the case of interdigitated implementation, there is only a good dispersion and symmetry between the devices by itself (just considering the diffusion and polysilicon layers), while for the metal interconnections there is not. The lengths of the metals used to interconnect the fingers of each transistor are very different; this causes the difference between the extracted values of capacitance and resistance. The devices of interdigitated implementation are more tolerant to gradient variations of temperature, process, stress, etc. due to the symmetry between them [Lomelí-Illescas-14] (simulation tests, not shown in this chapter, confirm these effects); however, the interconnections of the devices have a poor matching.



b)

c)

d)

e)

Fig. 3.10 Layout implementation for an array of 3 stacked NMOS transistors, each of them with 4 fingers: a) schematic diagram; b) shared diffusion placement implementation; c) shared diffusion implementation using all the metallization for interconnection; d) interdigitated layout placement implementation; e) interdigitated layout implementation using all the metallization for interconnection.

TABLE 3.3. SUMMARY OF LAYOUT IMPLEMENTATIONS OF AN NMOS ARRAY OF THREE STACKED DEVICES

| Topology | Metallization for interconnection | | | | | | Including only placement | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ra | $C_a$ | $R_b$ | $C_b$ | Rc | $C_c$ | Ra | $C_a$ | $R_b$ | $C_b$ | Rc | $C_c$ |
| | (Ω) | (pF) | (Ω) | (pF) | (Ω) | (pF) | (Ω) | (pF) | (Ω) | (pF) | (Ω) | (pF) |
| Shared diffusion | 1330 | 552 | 3110 | 900 | 3070 | 1 110 | 1030 | 266 | 2050 | 353 | 1920 | 500 |
| Interdigitated layout | 1780 | 983 | 2380 | 825 | 4360 | 1 540 | 1180 | 420 | 605 | 270 | 2350 | 457 |

In Fig. 3.10 are shown the layout implementations of the array of three NMOS stacked devices, using the two different levels of details and two topologies. Fig. 3.10b and Fig. 3.10c show the implementations for One Share Diffusion, including the metallization for interconnecting and considering only the placement, respectively. Fig. 3.10d and Fig. 3.10e show the Interdigitated Layout implementations, using the metallization for interconnecting and considering just the placement, respectively. As in the previous cases, all layout implementations are DRC clean and short-circuit clean; if the complete metallization is included, the layouts are also open-circuit clean.

## 3.8. Summary and Improvements

In this chapter, we described the general features of the proposed CAD tool for analog layout synthesis. We described the main stages and steps that form it, which are intended to speed up and improve the quality of the layout implementation of basic analog circuits. The layout placement is based on an internal database of fundamental structures, which allows the addition of new structures and topologies. The designer can select different levels of detail and different tool configurations. This information may help to reduce the number of circuit adjustments and prevent possible issues on final design stages.

Through some layout combinations, we aimed at testing the efficiency and utility of the proposed CAD tool for the generation of common analog structures. The resultant layout implementations were DRC clean for width, spacing and enclosure rules; all the implementations were short-circuit clean. For those cases where the level of detail includes metallization, the layouts were also open-circuit free. All generated layout views are DFM compliant and are ready to be included in more complex circuits or designs. The generation of each of these layouts required an execution time of less than a minute. To compare all these layouts views and to illustrate the information they provide, parasitic resistance and capacitance were extracted using a proprietary Intel Corporation extraction tool. The information provided by the extraction process on each layout helps to understand and determine how the non-idealities of the layout could affect the performance of a circuit. These experimental results show that this new layout tool is capable of producing high quality layouts comparable to those manually done by layout experts but with much less effort and design time.

All the layout implementations presented in the test cases were generated individually since the tool was manually configured by the designer for each new layout implementation. All the parameters for each new layout configuration were set separately; also, the extraction algorithm was executed manually over each of these layouts, and the extracted parasitic values were read from the report files of the algorithm. To ease the designer work, it would be desirable to include the option to set multiple configurations for the layout generation of the two structures presented. Also, add an option to execute the extraction algorithm automatically after the layout creation and use a parser scripting to obtain parasitic values automatically from the report files.

In the next chapter an enhanced version of the proposed CAD tool that incorporates some of the features commented above is presented. The primary purpose is the automatic creation of multiple layout versions of the two analog circuits under analysis, their characterization and the subsequent generation of a layout library.

# 4. CAD Tool for the Automatic Layout Generation of Libraries of Common Analog Structures

In the previous chapter, the development of a CAD tool that helps to speed up the layout implementation of analog structures/modules focused on differential pairs and arrays of series-connected transistors is described. A detailed description of this tool is also described in [Lomeli-Illescas-16]. This tool enables the implementation of robust and reliable layouts in a reduced amount of time.

One of the main features of the tool is the generation of multiple layout versions of the same circuit using different levels of detail; to come up with a close to optimal layout version, the tool allows users to input different layout configurations and setting options. 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. One disadvantage of the tool is that for the generation of each new layout version, the user has to set the different parameters individually and manually through the configuration GUI. For the characterization of these layouts, the extraction algorithm needs to be executed manually over each new layout, and the information of extracted parasitic values are read from the report files of the algorithm

In this chapters the description of a significantly enhanced version of layout automation tool presented in Chapter 3 is described; the improvements developed on the tool allows the automatic generation of multiple layout versions for the differential pair and stacked devices structures. Among the main characteristics already described in Chapter 3, some additional features are included, namely:

a) It allows the automatic generation of multiple layout versions for both fundamental structures and modules, varying parameters such as the number of transistors (for the case of stacked devices), the width of the devices, and the number of fingers of each component, 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) The layout versus schematic (LVS) verification flow described in [Lomeli-Illescas-16] is launched over all the elements of the database.

d) A list of valid and invalid layouts is automatically generated.

e) A summary report that includes parasitic elements information, total area, the total number of devices and rows, and a layout versus schematic (LVS) test results is also automatically created.

From the different layout solutions on the database, designers can choose the option that best meets their requirements. 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.

Most of the scripts that form the tool described in Chapter 3 are reused for the generation of the layout libraries; however, as expected some scripts are modified to enable this new feature.

In this chapter, a detailed qualitative description of the process followed by the tool, as well as the updates and modifications for the scripts are presented. As in [Lomeli-Illescas, 16]. The scripting language used is TCL, the tools used for the layout parasitic extraction and the LVS are internal tools from Intel Corporation and the technology used is a nanoscale Intel Corporation process.

## 4.1. CAD Tool General Description

As it was addressed in Section 3.13, this CAD tool is mostly a collection of scripts that allow the generation of multiple layout versions of a specific circuit for the subsequent the generation of their corresponding database. This database may allow the designers to obtain useful information for the optimal implementation of their circuits. This information could include 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.

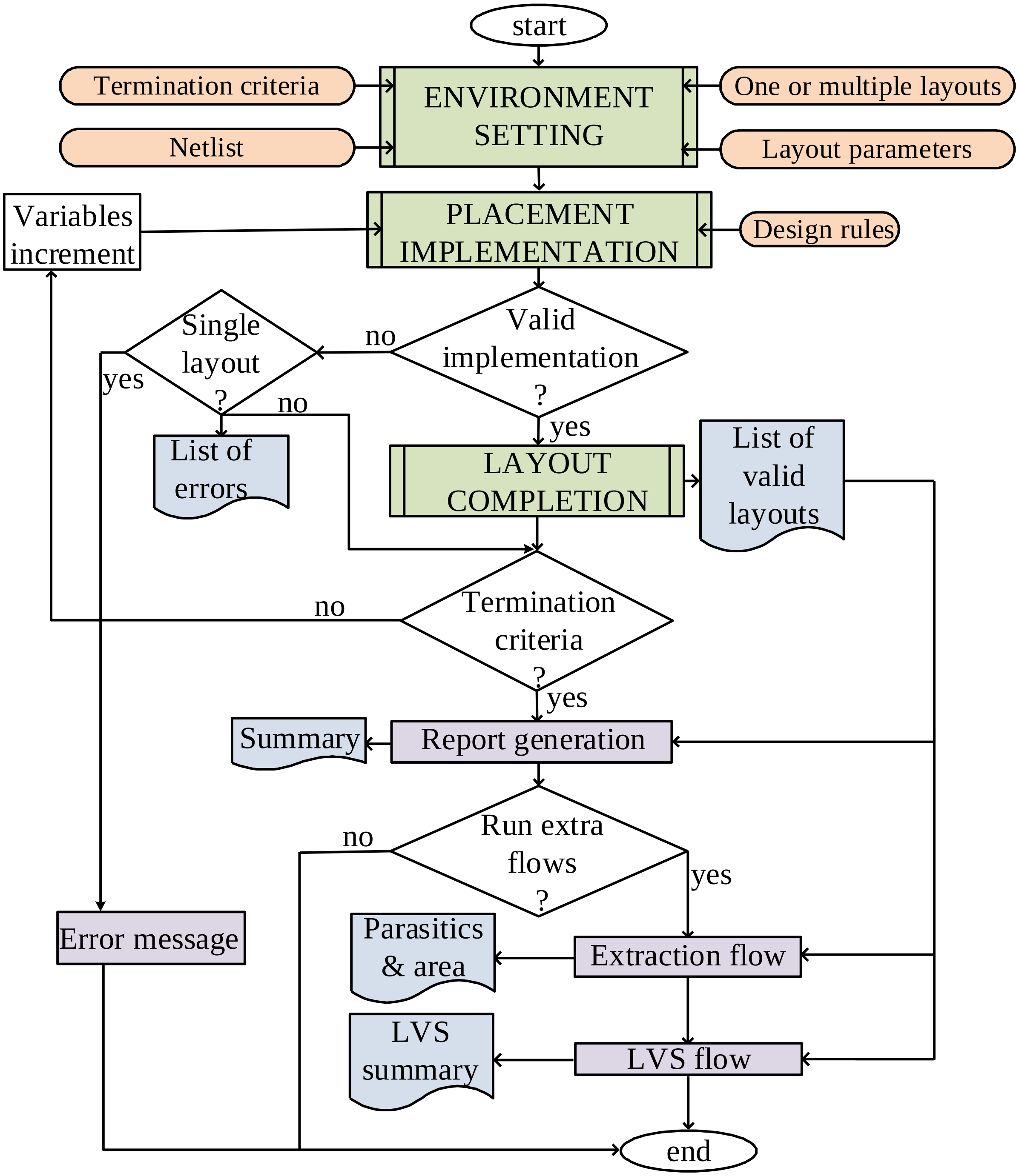


Fig. 4.1 Flow diagram of the proposed CAD tool for the automatic layout generation of libraries of common analog structures.)Figure taken from [Lomeli-Illescas 18].

The new version of the tool is formed by the same three sections described in previous chapters: environment setting, placement implementation, and layout completion.

The flow diagram of the CAD tool is shown in Fig. 4.1. It follows that one in Section 3.1 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. When a single layout is selected, users specify a netlist and the names of the transistors that form the structure to be implemented. When a database is created (for the option of multiple layouts), the user has to specify the termination criteria which can include the area, the width, and the height of the layout, or the maximum number of valid implementations.

Other set of inputs consists of the layout parameters (see Fig. 4.1), which 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 fringe dummy transistors , the number of rows and the insertion of substrate connections.

Placement Implementation is the same stage as the one used in [Lomeli-Illescas-16], but enabling the option to create multiple layouts; in this section, the devices are drawn one by one to form the required structure. Only diffusion P or N and polysilicon layers or base layers [Lomeli-Illescas-16] are included. Each transistor is drawn complying DRC rules (the DR file is loaded for this stage). Two new files are created, one for a new schematic view and another one for the layout view. A cleaning process is executed to delete any possible false layout elements. Next, the “Row Calculation” step 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 “variable increment” step. Then the placement-drawing step creates all the devices on the layout cell.

If a problem is detected during the placement implementation, this is reported in an error report file, indicating the reason for such problem. If no errors are found, the Layout Completion stage is executed; if a problem is found, this section is skipped, and a report containing the list of errors is generated (see Fig. 4.1). If an error is detected for a single layout case, the corresponding error message is displayed, and the CAD tool stops its execution.

The layout completion stage is shown in Fig. 4.1 is the same as that one presented in

[Lomeli-Illescas-16]. Here, the rest of the layout elements, such as metals, contacts, or ID[3] layers, are included in the implementation. Once all the layout elements have been added, the name of this structure with its corresponding characteristics is saved on a list of valid layouts.

Once the layout is completed, if the completion criteria have been reached, a report that contains all the information related to the implemented layouts is generated as well a summary file. If it is not the final implementation, i.e., if the termination criteria are not satisfied, then the "variable increment" stage updates the information to start a new placement.

Finally, the parasitics extraction is no longer a separate function. This process can be now executed automatically for all the different layouts implemented and saved in the database, following the list of valid layouts (see Fig. 4.1). Similarly, our tool can run an LVS test over all the created layouts. Each of these processes generates their corresponding report, including the parasitic elements information and indicating if the layouts are LVS clean or not.

In the following sections, a more detailed description of the main stages of the flow diagram in Fig. 4.1 is presented.

## 4.2. Environment Setting

As in [Lomeli-Illescas-17], in this stage, the user-defined specifications and parameters for the single or multiple (library) layout implementation are captured.

### 4.2.1 Main Window

The graphical user interface (GUI) employed in [Lomeli-Illescas-16] was modified to add the new options. Through this GUI, the user can select if a single layout or a layout database will be created. In the case of a single implementation, the user can load a netlist to choose the transistors that form the structure to be implemented or can merely select the structure 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

[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.

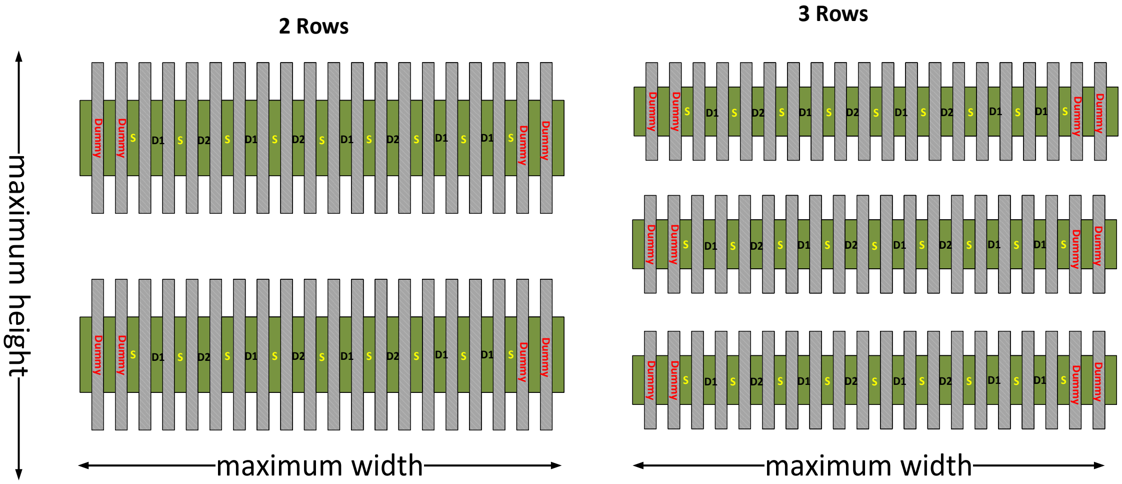
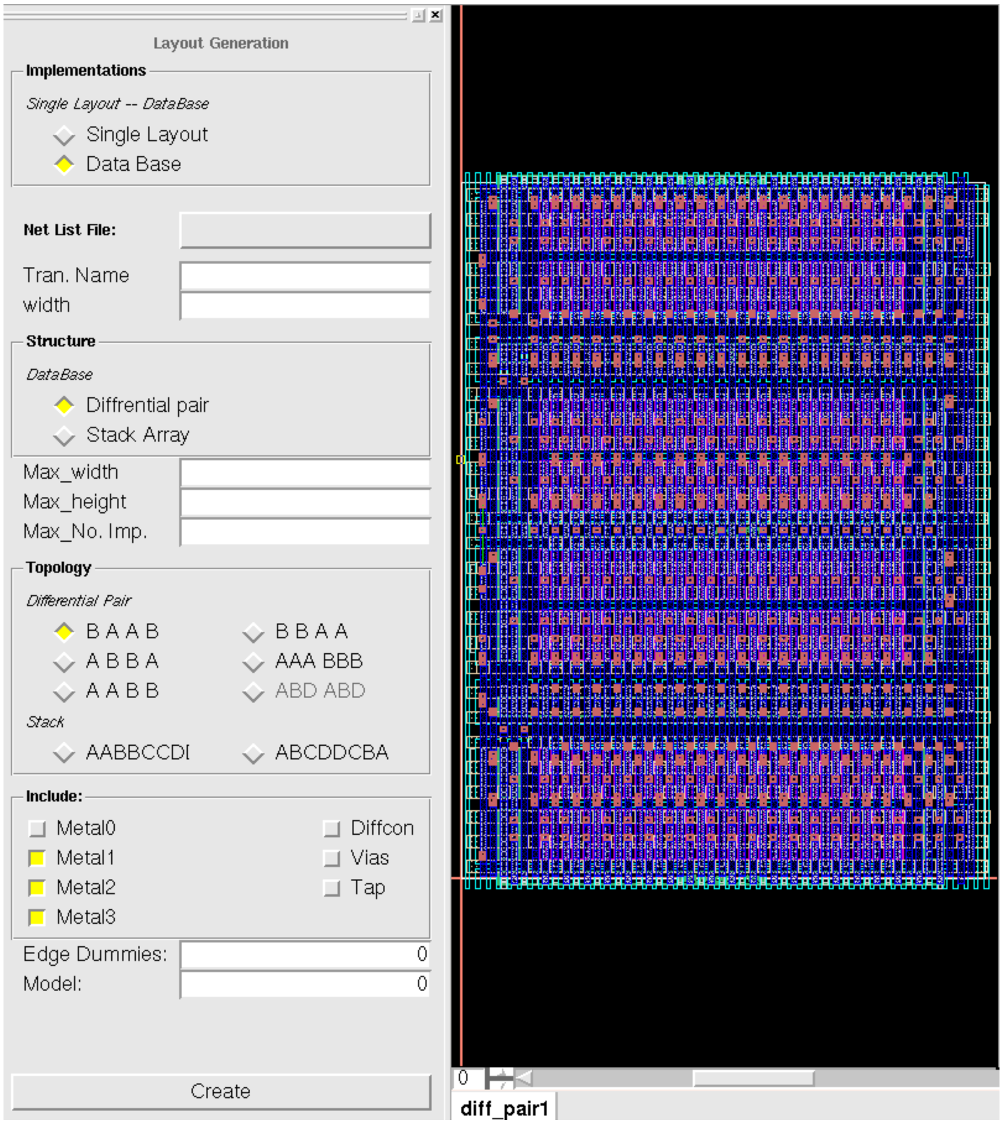


Fig. 4.2 Variation of the number of rows, in which the layout is implemented depending on the transistor's width.

one of the two available structures. From the netlist, the transistor's dimension (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. When the database option is selected, termination criteria are needed for the creation of a database. The termination criteria are the maximum number of layout implementations, the maximum width value, and the maximum height value. The maximum allowed width delimits the maximum number of transistors in a single row, while the maximum height limits the maximum number of rows; the maximum possible number of rows to be used is also limited by the width of the devices, as we can see in Fig. 4.2. In this figure, we can notice that for a specific width, the layout can be implemented in three rows, while for larger transistors the layout can only be implemented in two rows.

In the tool proposed in [Lomeli-Illescas-16], the user can define the level of detail for the layout: placement, base layers, metallization for interconnection, or DFM compliance (adding dummy fills, guard rings, etc.). This option is also enabled in this CAD tool; however, it is highly recommended to use the last two levels of detail (including at least all the metal for interconnection or also the DFM compliance elements) and to improve the accuracy of the information obtained from the layout implementation.

Using the GUI, users can also define if the algorithm of parasitics extraction and the LVS verification flow 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. 4.3.



Layout Generation

**Implementations**

*Single Layout -- DataBase*

○ Single Layout

● Data Base

**Net List File:**

Tran. Name

width

**Structure**

*DataBase*

● Diffrential pair

○ Stack Array

Max_width

Max_height

Max_No. Imp.

**Topology**

*Differential Pair*

● B A A B ○ B B A A

○ A B B A ○ AAA BBB

○ A A B B ○ ABD ABD

*Stack*

○ AABBCCDI ○ ABCDDCBA

**Include:**

□ Metal0 □ Diffcon

☑ Metal1 □ Vias

☑ Metal2 □ Tap

☑ Metal3

Edge Dummies: 0

Model: 0

Fig. 4.3 Sample image of the proposed tool with a generated layout

## 4.3. Placement Implementation

Once the termination criteria and the different layout’s parameters have been defined, the next step is the creation of the multiple layout versions. In the original proposal, only one layout is created and saved by this process. In this proposal, the placement implementation algorithm is executed uninterruptedly until one of the completion criteria is reached. The considerations needed for the generation of the layout, due to the technology process are the same as described in Layout Considerations Subsection 3.5.1.

### 4.3.1 Cell Creation and Cleaning

As in [Lomeli-Illescas-16], two new files are created, one for a new schematic view and another one for the layout view. For this proposal, the new schematic view is based on the layout that is being implemented. The schematic view includes only the devices and their corresponding connections. The name of the file is unique and uses the same format: a reference of the structure to be implemented (diff_pair or stack) and an index number (which guarantee unique names). Once the new files are created, the next stage is to clean any possible layout element that by mistake could be created on the new cells; ideally, the cell should be clean, but it is necessary a cleaning process to delete any possible false layout element. The next step now is the creation of the layout itself, starting with the placement of the devices.

### 4.3.2 Parameters Modification

The parameters that our tool modifies for the generation of layouts are the number of fingers/devices and their widths. In the case of the stacked devices module, the number of transistors that form the array is also modified. The CAD tool increases the number of fingers and finger's width, as long as the layout can fit in the available area (termination criteria); it also distributes the devices into different rows as necessary. The maximum number of fingers and the maximum width value depend on the available area.

Since the proposed tool modifies the number of fingers and the finger's width, it is possible that different implementations have the same effective width, defined as $W_{\text{eff}}$ = transistor's finger's width × number of fingers, but a different number of fingers and a different layout placement, this is illustrated in Fig. 4.4. In Fig 4a the transistor A is $W$ wide. In Fig. 4.4b the transistor A is broken into four smaller transistors with a width of ¼$W$. Another option is to split the transistor A in two smaller transistors of ½$W$ and combine them as in Fig. 4.4; finally, we can use two transistors of two fingers of ¼$W$ width but placed in two different rows as in Fig. 4.4. If the terminals of all the devices are correctly connected, the three implementations should have the same $W_{\text{eff}}$ width.

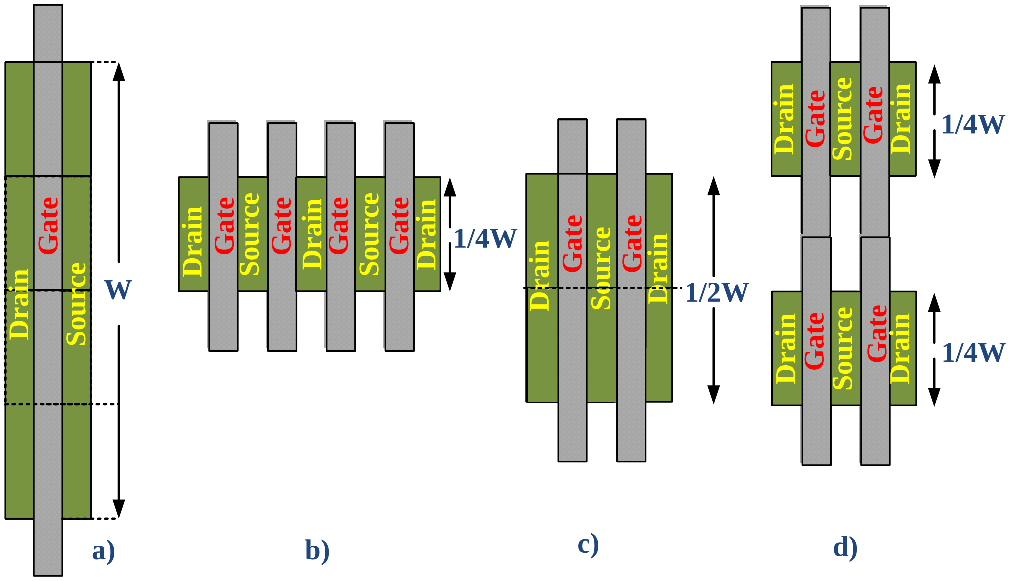


Fig. 4.4 Example of transistors finger: a) Transistor A is *W* wide and uses one finger; b) Transistor A is divided into 4 fingers with a width of ¼*W*; c) Transistor A can be divided into 2 fingers with a width of ½W; d) Transistor A is split into two transistors of two fingers of ¼*W* width but placed in two different rows.

### 4.3.3 Row Calculation and Placement Drawing

For the placement drawing function, the scripts used by the current tool are the same group of scripts used in [Lomeli-Illescas-16]; the only new added function is the “Row Calculation” one, which is responsible for calculating the number of rows that are required for the layout implementation. This calculation is based on the values of the transistors dimensions, the number of fingers, and the number of transistors (if required), which are updated in each iteration; it also depends on the maximum width and height values allowed. Once the information relative to the number of rows and devices has been updated, the scripts that form the placement generator process are executed:

a) Validation of 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 is 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) 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: here, the devices are created individually, following the order defined in the previous block. The Transistor Generator algorithm creates every single transistor by drawing it on a specific location. The Placement Drawing control sub-function is responsible for indicating where to draw each device. The layout draws transistor by transistor and row by row, following the order defined by the Topology Setting function.

### 4.3.4 Errors Report

After all of these scripts have been executed, the next step is to include this layout in the error report file or the list of valid layouts. If the" Validation of Number of Transistors" function reports an invalid implementation, it is included in an errors report file, where the problem is described, indicating the reason of the error and the specification of that topology. Additionally, if an error is found during the placement drawing, this is also reported and included in the errors file. As mentioned before, in the case of single layout implementation, this information is not included in the errors report file but also shown on a message window, and the CAD tool stops its execution. In the case of the database creation (multiple layouts), the CAD tool does not stop when an error is detected and saved. If no error is found in the implementation or the validation of the number of devices, the layout is included in a document with the list of valid layouts and the layout completion section is executed over it; if a problem is found, the layout completion section is skipped for that case.

## 4.4. Layout Completion

This process is the same as the one presented in [Lomeli-Illescas-16]. Here, all the metal routes and contacts are added, depending on the level of detail selected. 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.

As we commented before, it is recommended to include at least the base layers or the metals for interconnections. The addition of contacts or vias is no longer optional, and they are included at the higher possible level. As in [Lomeli-Illescas-16], the tool is still limited to M2 for horizontal routing and M3 for vertical routing, however, higher metal layers can be included if they are needed for more complex structures.

Once the layout is completed, the name of this structure with its corresponding characteristics are saved on a list of valid layouts (see Fig. 4.1).

### 4.4.1 Completion Criteria, Database and Report Generation

Once the layout is completed, it is necessary to verify if the tool has reached the completion criteria; this means checking if the tool completed the maximum number of layout implementations, or if the final layout reached the maximum width and height. If it is the case, the next step is the generation of a report that indicates which layouts were created, their names and their characteristics, as well as the library that they belong to; also, a document that summarizes this information is created. If the termination criteria are not satisfied, the "variable increment" step is responsible for updating the values for the creation of a new layout; starting from the number of fingers of each device and then the size of the transistors; also the number of devices is updated (in the case of the array of stacked devices). This information is updated on the placement implementation section to start the creation of a new layout.

## 4.5. Extra Flows

Once the termination criteria are satisfied, and the reports are generated, the extraction algorithm and LVS flows can be executed over all the elements saved in the database following the list of valid layouts; if this option is selected a parasitic extraction process and the LVS verification flow are executed over each layout.

### 4.5.1 Extraction

As mentioned before, the algorithm used to perform the parasitic extraction is that one in [Lomeli-Illescas-16], which is an internal design tool from Intel Corporation.

All the information relative to the parasitics elements (resistance and capacitance in the current version) is obtained from the report files that the extraction algorithm generates. When a single layout is created, 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. Our proposed tool filters out the data of these reports to obtain the information related to some specific parameter. For instance, the tool 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, our tool 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. In this current version, the only value obtained is the sum of capacitance to VSS. Once the extraction process finishes for all the layouts, a report is generated, indicating the topology, the value of the capacitances, and the required area.

### 4.5.2 LVS Flow

This flow runs over all of the layouts saved in the database to detect any possible problem with the layout implementation. It only verifies that the layouts are open and short-circuit clean. Ideally, all the implementations should be clean, but this flow ensures so, such that the designers can use the created layouts in the implementation of their circuits. This verification flow can also be used for debugging the tool, in case one layout was not correctly implemented.

Once the LVS flow checked all the layouts included in the database, 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 flagged. For our current version, only LVS flow is included; other flows can be added into the tool such as DRC or density checkers, but it is not recommended since these are usually run over structures that are more complex.

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

## 4.6. Test Cases

In this section, some examples of the analysis of the reports that are generated by the CAD tool are presented. The examples are intended to show some of the features and capabilities of the proposed CAD tool. For the analysis, four libraries are generated: two for the differential pair and two for the array of stacked devices. The libraries are created using two different topologies: the Interdigitated Layout (IL) and the One Shared Diffusion (OSD). The parasitics extraction algorithm is executed over all the elements saved in the databases, while the reports generated by this are used to compare the different implementations. The information from these reports is used for a comparison of a specific parasitic value ($C_{vss}$) and the required implementation area for all the different layouts in the databases when different parameters such as the number of fingers/devices, finger's width, and the number of transistors in the arrays are considered. This information is presented in four different plots for better clarity.

### 4.6.1 Library Generation

For the new libraries, the termination criteria are set up using values that enable the creation of the layouts using from one up to three rows with thirty as the maximum number of devices in a row (including the dummy devices, which are two at each side of the array) for its implementation. The maximum limit for these two criteria is illustrated Fig. 4.5. The selection of these values has the intention to test the different features of the CAD tool, including its capability to distribute the transistors uniformly among a different number of rows, addressing those cases where a given distribution is not possible.

As it was addressed previously, the dimensions used for each transistor are discrete values that enable the placement of an even number of contacts (or vias) to avoid Reliability Verification (RV) problems. The finger's widths used for the transistors' implementation are two, four, or six times the minimum width, $W_{\min}$.

The "Variable increment" step described in the previous subsection is used for updating the values for the creation of a new layout. For the differential pair, the process starts from the

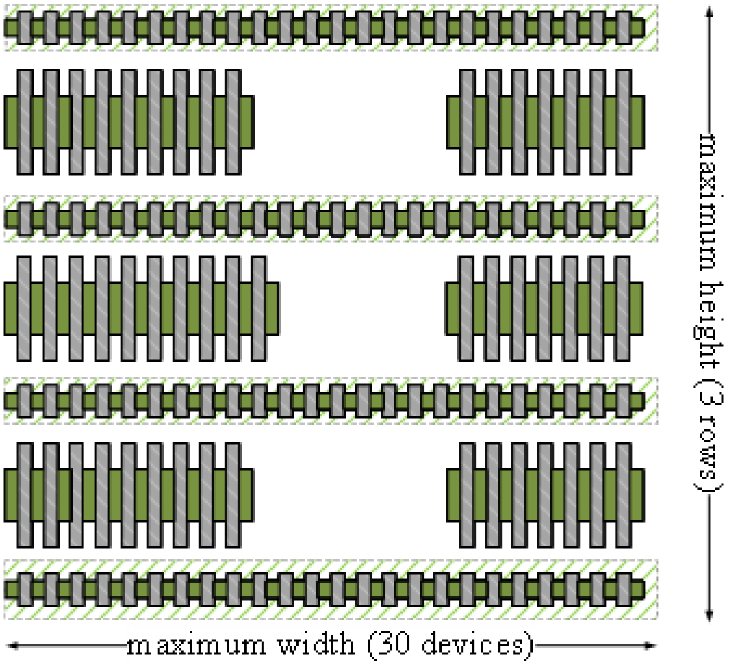


Fig. 4.5 Illustration of the area used for the implementation of a sample layout included in the generated libraries

number of fingers of each device and then the size of the transistors. For the array of stacked devices, the first variable to increment is the number of devices in the array starting from two to five, then the number of fingers of each device, and finally the size of the transistors. The maximum number of devices in the array is defined as five since as it was addressed in [Lomeli-Illescas 16], the addition of more devices does not contribute significantly to the performance of the circuit.

Table 4.1 shows the results for the generation of the four libraries. We can notice that there are more valid implementations for the libraries of the array of stacked devices than for the library of the differential pair; this 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 the elements uniformly between different rows.

If we compare the libraries of the differential pair structure, for the one that uses the OSD topology, more layouts are generated; this is because, as it was commented in [Lomeli Illescas 16] for this topology, there is no limitation in the number of fingers, since they can be an odd or an even number. A similar situation occurs in the case of the stacked array libraries: since the IL topology does not have restrictions for the number of fingers, more valid layouts can be

TABLE 4.1. SUMMARY OF RESULTS FOR THE GENERATED LIBRARIES

| Parameter Evaluated | Differential Pair (IL) | Differential Pair (OSD) | Stacked Devices Array (IL) | Stacked Devices Array (OSD) |
|---|---|---|---|---|
| Number of valid layouts | 19 | 38 | 96 | 46 |
| Number of invalid layouts | 3 | 7 | 22 | 12 |
| Percentage of valid layouts | 84% | 81% | 77% | 73% |
| Maximum capacitance to VSS | 327 $C_{min}$ | 248 $C_{min}$ | 458$C_{min}$ | 303 $C_{min}$ |
| Maxim number of devices | 90 | 90 | 90 | 77 |
| Maximum number of rows | 3 | 3 | 3 | 3 |
| Average implementation time per layout | 98 seconds | 108 seconds | 104 seconds | 113 seconds |
| Average extraction time | 423 seconds | 453 seconds | 480 seconds | 512 seconds |

implemented using this topology that the OSD one.

Regarding the average implementation time per layout, it can be noticed that for all the libraries 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. The time needed for the creation of libraries for stacked devices is longer than the time required for the creation of libraries for the differential pair. In addition, the time required for creating libraries with IL topologies is longer than the time for creating libraries using OSD topology; this is because IL topology in nature is a more complex structure then its placement and routing are more difficult.

From Table 4.1, it is noticed that for the total capacitance to VSS node, the complexity of the routing stacked devices structures increases their total capacitance compared to those of the differential pair (a more in-depth analysis is presented in the next subsection). Finally, the structures that used the OSD topology have larger $C_{vss}$ values than they counterpart that used the IL topology; this is described in more detail in the next section.

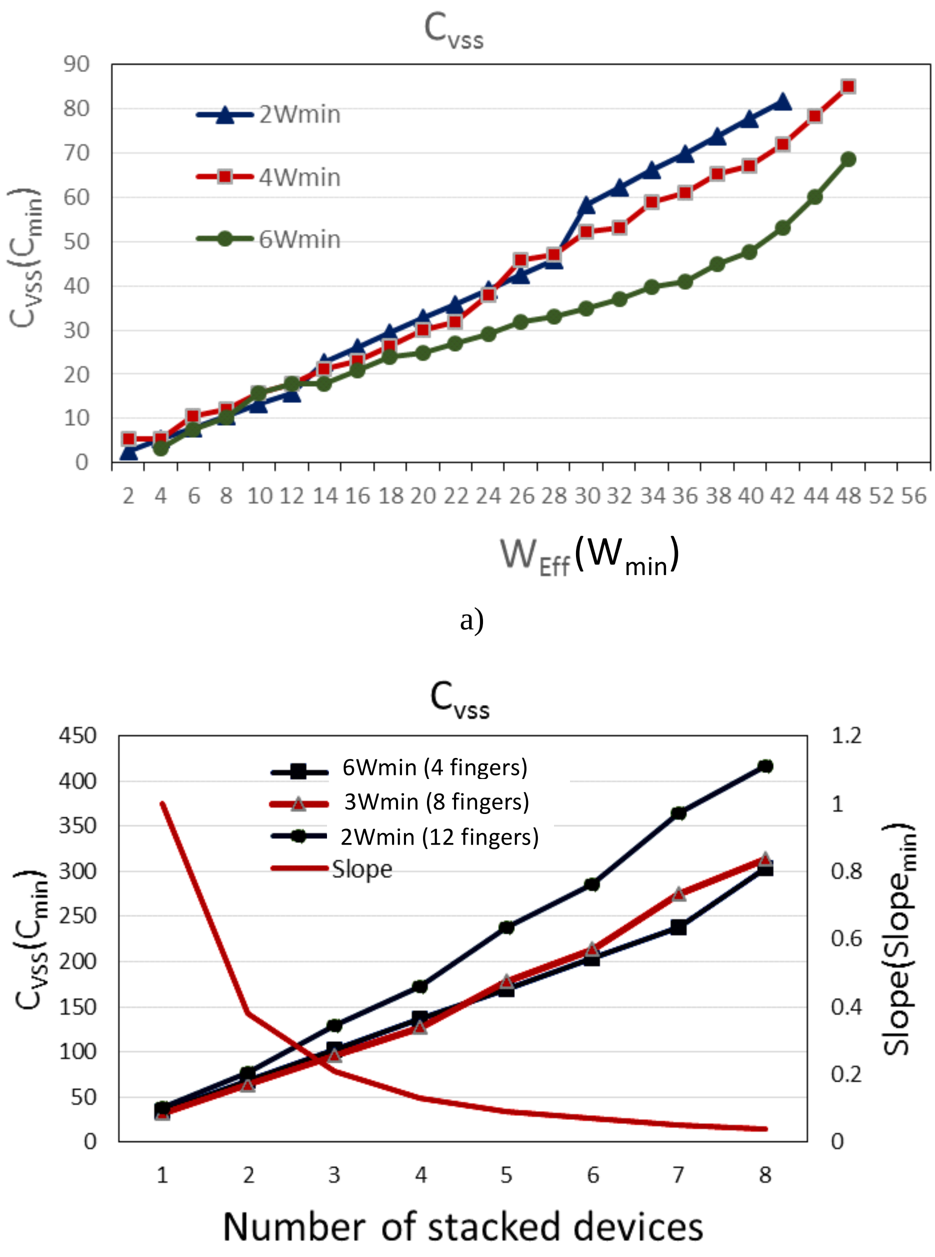
### 4.6.2 Parasitics Analysis

An analysis of different layout implementations of an array of stacked transistors is presented in [Lomelí-Illescas-17], comparing the tradeoffs between different topologies and different layout parameters in terms of their parasitic elements. The layout implementations used for that analysis are generated using the automatic synthesis tool in [Lomelí-Illescas-16]. A similar process is performed here to show the benefits of using enhanced CAD tool. The algorithm for parasitics extraction is executed over all the valid layouts generated. The $C_{\text{vss}}$ value and the area for all the valid implementations of the four databases are obtained and automatically organized for comparison between all the different layouts. These capabilities illustrate the information that our CAD tool can provide to the designers.

Fig. 4.6 and Fig. 4.7 show four 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 width, , and the number of transistors in the arrays. In all these cases the $C_{vss}$ values are expressed in terms of $C_{min}$ (the $C_{\text{vss}}$ capacitance for a single transistor using minimum dimensions).

Fig. 4.6a shows $C_{vss}$ values for the differential pair structure when different effective widths are used. The topology used is the IL one. For this example, and for illustration purposes, we parsed from the report file the cases for finger's widths equal to $2W_{\text{min}}$, $4W_{\text{min}}$, and $6W_{\text{min}}$. Naturally, as $W_{\text{eff}}$ increases, $\text{C}_{\text{vss}}$ increases as well. However, in Fig. 6a it is observed that for large $W_{\text{eff}}$ (in this example larger than $14W_{\text{min}}$) the capacitance value tends to be smaller when the finger's width is larger (or when less fingers are used). This indicates that 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. Additionally, as we use more fingers, more rows are required for their placement; this increases the number of interconnections and their length, as well as the capacitance associated to them.

Fig. 4.6b shows $C_{\text{vss}}$ values when a different number of stacked devices and fingers are used on an array, but keeping $W_{\text{eff}} = 24W_{\text{min}}$ for all cases. In this example, the topology used is the OSD. The cases parsed from the reports are for a finger's width equal to $2W_{\text{min}}$, $3W_{\text{min}}$, and $6W_{\text{min}}$ (which correspond to 12, 8 and 3 fingers respectively).As in the previous case, when larger finger's widths are used, the $C_{\text{vss}}$ 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 metal wires required. In consequence,



a)

b)

Fig. 4.6 Comparison of the Cvss capacitance value for: a) differential pair using different Weff and varying the transistor width; b) array of stacked devices using a different number of transistors and varying the number of fingers.

the area necessary for layout implementation is larger when transistors are divided into fingers than when large finger’s widths are used, as it was confirmed in [Lomeli Illescas 16]. 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. 4.6b 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 [Lomelí-Illescas-17]). In summary, Fig. 4.6b illustrates the combined effects that different layout implementations have over total parasitic values and output impedance.

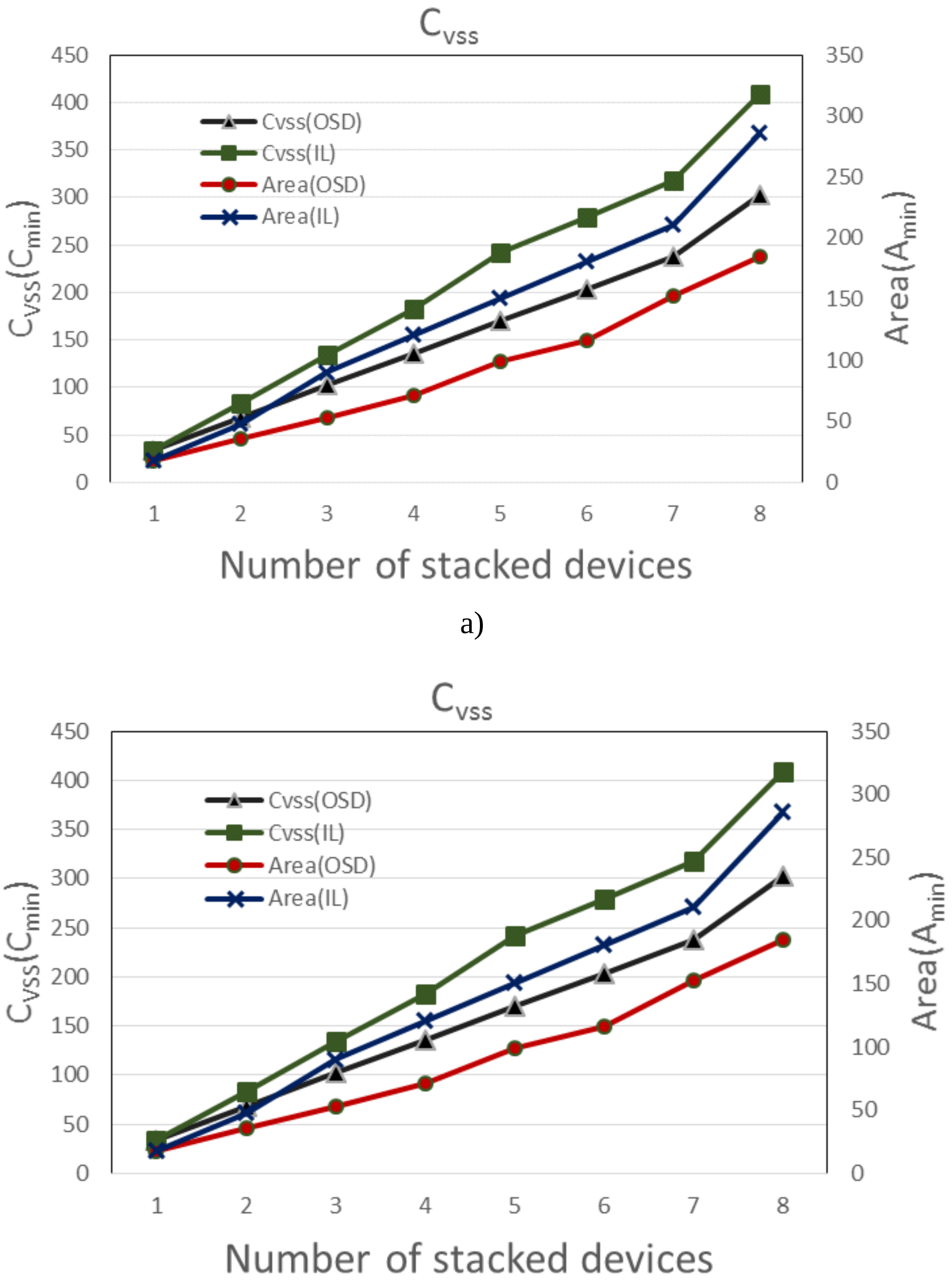
Fig. 4.7a shows $C_{\mathrm{vss}}$ values and the required layout area varying the number of fingers for a differential pair structure but also considering the two different topologies: IL and OSD. In all cases, the finger's width is $2W_{\mathrm{min}}$. Naturally, the topology determines the length 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. 4.7a. Since, the OSD topology requires less and shorter metal wires, the values of its parasitic elements are smaller than those for the IL topology (see Fig. 4.7a). Similarly, since the OSD topology employs fewer wires than the IL topology, it requires smaller layout implementation areas (see Fig. 4.7a).

Finally, Fig. 4.7b shows $C_{\mathrm{vss}}$ values and the required layout area varying the number of stacked devices on the array structure also considering the two different topologies: IL and OSD. For both topologies, the finger's width is $2W_{\mathrm{min}}$, and the number of fingers is 3. As in the previous cases, the additional metal interconnection wires increase the values of the $C_{\mathrm{vss}}$ and the required implementation area for the case of the IL topology.

The above graphical examples also illustrate some of the features and capabilities of the proposed CAD tool.

## 4.7. Summary

In this chapter, we addressed the main features of an analog layout synthesis tool. This tool is intended to improve the layout implementation of fundamental analog circuits as it enables the creation of layout libraries. We described the main stages and functions that form it and that are used for the generation of the analog layout libraries.



a)

b)

Fig. 4.7 Comparison of the $C_{vss}$ capacitance value for using two different topologies: the interdigitated layout (IL) and the one shared diffusion (OSD): a) deferential pair; b) array of stacked devices.

The results and the analyses executed using the reports generated by the parasitics extraction process show the information that these reports can provide to the designer. When multiple layouts are created, and the extraction process is executed over all of them, the analysis of all the reports can be very time consuming and errors prone. The CAD tool presented can filter and organize this information automatically, which enables designers to conduct more efficient analyses. The different tests executed show that this new layout tool is capable of producing many layout versions, identifying problems in their implementation, generating reports, and organizing this information in a short amount of time, which helps to reduce the design effort for analog circuits.

# General Conclusions

The development of an analog layout synthesis tool intended to accelerate the layout implementation of fundamental analog circuits and that enables the generation of layout databases is presented in this work. This tool offers the possibility of automatically creating multiple layout versions of two commonly used analog structures: the differential pair and arrays of series-connected or stacked devices. However, it can be extended to include other structures since the layout creation is guided by pattern definition templates, included on an internal database; the use of a systematic codification facilitates the addition of new topologies or analog structures.

The tool validates the possible implementations according to the required number of devices and rows, generating a list of valid and invalid layouts. The list of invalid layouts is very useful since it enables designers to identify structures that should not be used in their circuit implementations, even if they can be implemented at the circuit level. On another hand, all valid layouts are saved on a database, with the information of their characteristics; using this database the designers, can analyze this information and select the best option for a specific implementation. Additionally, the extraction process that can be optionally executed over all the layout views saved in the database provides extra data about the layout implications, characteristics, and tradeoffs.

The tests that are presented in this work illustrated the effectiveness and utility of the proposed CAD tool for generating multiple layout versions of analog structures integrated into libraries. Also, the multiple reports that were generated exemplify the information that our CAD tool can provide to the designers.

In Chapter 1, some of the adverse effects, tradeoffs, and challenges of the implementation on an analog layout in current nanoscale technologies were described, while in Chapter 2 a detailed analysis of the effects and the importance of using stacked devices structures in this kind of circuits is presented. These analyses pointed out the necessity to have a CAD tool that facilities a fast and efficient implementation of analog layout structures while providing useful information for the designers relative to the characteristics of such implementations. Based on this, in Chapter 2 the first version of our CAD tool is presented, this tool allowed the automatic generation of the analog structures commented before, but individually since for each new layout the tool needs to be manually configured. The tool facilitates the parametric generation of multiple layout versions of analogs structures, using different configurations and different levels of detail. The generation of multiple layouts allows the analysis of the tradeoffs between different topologies and different layout parameters in terms of their parasitic elements.

The advantages of the generation of parametric layouts motivate the enhancement of the CAD tool, to enabling the automatic creation not only of a single layout but a database of the aforesaid analog circuit. This tool generates multiple layout versions for both fundamental structures, varying parameters such as the number of transistors, the width of the devices, and the number of fingers of each component for the creation of database or library. All the layouts are short-circuit clean and design rules checker (DRC) compliant; the list of valid and invalid layouts is generated, as well as a summary of results generated by the tool provide useful information to designers, that they can process or analyze to make a performance comparisons and select the alternative that best meets their requirements.

This thesis offers the possibility of a number of future research opportunities: for the generation of a database our tool creates all the possible combinations for a specific termination criterion (in this case a limited area); this could require a long execution time and a high computational cost, depending on the total number of implementations. This problem is aggravated if the extraction algorithm is run over all of them. So, it would be desirable that the tool should be capable of generating specific layouts that meet pre-defined criteria, not only related to the area but other constraints; also the tool should be capable of finding an optimal implementation according to the constraints defined by the user.

Since the tool is capable of producing a large number of layouts in a short time one of the optimization methods that can be implemented is the space mapping technique. This allows the generation of a coarse model of the layout that can provide information about the layout characteristic and parasitic elements in a shorter period of time than the traditional extraction algorithm.

# Conclusiones Generales

En este trabajo se presenta el desarrollo de una herramienta de síntesis para *layout* analógico cuya finalidad es acelerar la implementación, física, *layout*, de estructuras analógicas fundamentales así como permitir la generación de librerías de dichos diseños. Esta herramienta ofrece la posibilidad de crear automáticamente múltiples versiones de diseño de dos de las estructuras analógicas más comúnmente utilizadas: el par diferencial y los arreglos de dispositivos conectados en serie o apilados, pero estas opciones pueden extenderse para incluir otras estructuras, ya que la creación de los diferentes *layouts* es guiada por plantillas predeterminadas para la ubicación de los dispositivos, las cuales están incluidas en una base de datos interna de la herramienta; el uso de una codificación sistemática para dichas platillas facilita la adición de nuevas topologías y/o estructuras analógicas.

La herramienta cuanta con la función de validar las posibles implementaciones de acuerdo con el número requerido de dispositivos y filas para su implementación, generando una lista de diseños válidos y no válidos. La lista de diseños no válidos es muy útil ya que permite a los diseñadores identificar estructuras que no son posibles de implementar físicamente a pesar de que puedan ser utilizadas en simulaciones o circuitos a nivel esquemático. Mientras tanto los diseños válidos se guardan en una base de datos, con la información de sus características, de esta forma los diseñadores, pueden revisar y analizar de esta información y seleccionar la mejor opción para una implementación o aplicación específica. Adicionalmente, el proceso de extracción que se puede ejecutar opcionalmente sobre todos los diseño guardados en la base de datos puede proporcionar información adicional sobre las implicaciones que la implementación física del circuito tiene sobre sus características y los compromisos que genera.

Los diferentes experimentos que se presentan en este trabajo ilustran la eficacia y la utilidad de la herramienta CAD propuesta para generar múltiples versiones de *layout* de estructuras analógicas, y después ser integradas en bibliotecas. Además, los múltiples reportes que se generaron ejemplifican la información que nuestra herramienta de CAD puede proporcionar a los diseñadores.

En el capítulo1, se describieron algunos de los efectos, desafíos y compromisos que se tienen en la implementación de *layouts* analógicos en tecnologías nanométricas actuales. Estos análisis muestran la necesidad e importancia de tener una herramienta CAD que facilite la implementación rápida y eficiente de estructuras analógicas, a la vez que sea capaz de generar información útil para los diseñadores en relación con las características de tales implementaciones.

Basado en lo anterior en el capítulo 3 se presenta la primera versión de nuestra herramienta CAD, dicha herramienta permite la generación automática de las estructuras analógicas comentadas anteriormente, pero de forma individual, ya que para cada nuevo diseño, la herramienta debe configurarse manualmente. La herramienta facilita la generación paramétrica de múltiples versiones de *layout* de estructuras analógicas, usando diferentes configuraciones y diferentes niveles de detalle. La generación de múltiples *layouts* permite hacer la comparación de los diferentes compromisos entre las diferentes topologías y parámetros de “layout” en términos de sus elementos parásitos.

Las ventajas que tienen la generación paramétrica de *layout* motivaron que se desarrollaran diversas mejoras a la herramienta de CAD, permitiendo la creación automática no solo de un *layout* individual, sino de una librería completa de las estructuras antes mencionadas. Esta herramienta genera múltiples versiones de *layout* para ambas estructuras, modificando distintos parámetros del mismo tales como la cantidad de transistores, el ancho de los dispositivos y el número de dedos de cada componente; todos los diseños se implementan sin ningún error de cortocircuito y cumpliendo con las distintas reglas de diseño (DRC). La lista de diseños válidos e inválidos, así como el resumen de los circuitos generados por la herramienta, proporcionan información útil a los diseñadores, que estos pueden analizar y realizar una comparación de su desempeño y seleccionarla opción que satisfaga de mejor manara sus requerimientos;

Esta disertación ofrece distintas posibilidades y oportunidades para futuras áreas de investigación: para la creación de una base de datos, nuestra herramienta crea todas las combinaciones posibles de acuerdo a un determinado criterio de terminación (en este caso, un área determinada); dependiendo del número total de implementaciones, la generación de todas ellas podría requerir un tiempo de ejecución demasiado largo y un alto costo computacional. Este problema se agrava si el algoritmo de extracción se ejecuta sobre todas esas implementaciones. Por lo tanto, sería conveniente que nuestra herramienta se capaz de generar *layouts* específicos que cumplan con otros criterios predefinidos, no solo relacionados con el área sino con otras restricciones; también la herramienta debería ser capaz de encontrar una implementación óptima de acuerdo restricciones previamente definidas por el usuario.

Dado que la herramienta es capaz de producir múltiples *layouts* en un corto período de tiempo, una de las técnicas de optimización que se puede implementar es la “mapeo espacial”; Esta técnica permitirá la generación de un modelo de *“layout”* burdo, que puede proporcionar información sobre las características de los *layouts* y sus elementos parásitos, en un período de tiempo más corto que el algoritmo de extracción tradicional. Esta opción podría ser explorada en trabajos futuros.

# Appendix

# A. LIST OF INTERNAL RESEARCH REPORTS

1) I. Lomelí-Illescas and S. A. Solis-Bustos, “Analysis of the challenges in layout of analog integrated circuits and in the implementation of digital PLLS,” Internal Report *PhDEngScITESO-13-03-R*, ITESO, Tlaquepaque, Mexico, Dec. 2013.

2) I. Lomelí-Illescas and S. A. Solis-Bustos, “Analysis of layout challenges for subnanometric technologies,” Internal Report *PhDEngScITESO-14-16-R*, ITESO, Tlaquepaque, Mexico, Dec. 2014.

3) I. Lomelí-Illescas and S. A. Solis-Bustos, “Efficient layout techniques for subnanometric technologies,” Internal Report *PhDEngScITESO-15-14-R*, ITESO, Tlaquepaque, Mexico, Dec. 2015.

4) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Conceptual definition of a CAD tool for the automatic layout generation of common analog structures,” Internal Report *PhDEngScITESO-16-02-R*, ITESO, Tlaquepaque, Mexico, Jan. 2016.

5) I. Lomelí-Illescas, S. A. Solis-Bustos, V. Martínez, and J. E. Rayas-Sánchez, “Preliminary synthesis tool for automatic layout generation of common analog structures,” Internal Report *PhDEngScITESO-16-06-R*, ITESO, Tlaquepaque, Mexico, May 2016.

6) I. Lomelí-Illescas, S. A. Solis-Bustos, V. Martínez, and J. E. Rayas-Sánchez, “Test cases for a synthesis tool for automatic layout generation of common analog structures,” Internal Report *PhDEngScITESO-16-10-R*, ITESO, Tlaquepaque, Mexico, Jul. 2016.

7) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Analysis of the implications of stacked devices in nanoscale technologies for analog applications,” Internal Report *PhDEngScITESO-16-27-R*, ITESO, Tlaquepaque, Mexico, Dec. 2016.

8) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “analysis and evaluation of different layout implementations of stacked transistors,” Internal Report *PhDEngScITESO-17-07-R*, ITESO, Tlaquepaque, Mexico, May. 2017.

9) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Conceptual definition of a CAD tool for the automatic layout generation of libraries of common analog structures,” Internal Report *PhDEngScITESO-17-19-R*, ITESO, Tlaquepaque, Mexico, Jun. 2017.

10) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Test cases for a cad tool for the automatic layout generation of libraries of common analog structures,” Internal Report *PhDEngScITESO-17-27-R*, ITESO, Tlaquepaque, Mexico, Jun. 2017.

11) I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “New test cases for a cad tool for the automatic layout generation of libraries of common analog structures,” Internal Report *PhDEngScITESO-18-03-R*, ITESO, Tlaquepaque, Mexico, Jun. 2017.

# B. LIST OF PUBLICATIONS

## B.1 Conferences Papers

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. (ISBN: 978-1-5090-2532-9; e-ISBN: 978-1-5090-2533-6; INSPEC: 16650408; DOI: 10.1109/ANDESCON.2016.7836218).

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)*, Bogotá, Colombia, Mar. 2017, pp. 1-4. (ISSN: 2373-0862; ISBN: 978-1-5386-0416-8; e-ISBN: 978-1-5386-0415-1; INSPEC: 16837112; DOI: 10.1109/LATW.2017.7906750).

## B.2 Journal Papers

I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “A tool for the automatic generation and analysis of regular analog layout modules,” *Elsevier Integration - the VLSI Journal*, vol. *, no. **, pp. **, *** 2018. (p-ISSN: 0167-9260; published online: 30 Nov. 2018; DOI: https://doi.org/10.1016/j.vlsi.2018.11.005; regular publication pending).

# C. EXAMPLES OF ERROR REPORT FILES

## C.1 List of non-valid layout for array of stacked devices structure

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x                                               x
x             LIST OF NON-VALID LAYOUTS               x
x                                               x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x Confidential    x
x----------------------------------------------------x
x This file contains a list of non-valid layouts x
x                                               x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


Today's date = Wed sep 6 11:49:28
Machine = plxc25959

Topology: Array of Stacked Devices

Layout Library path: /p/acd/proj/work/ade/ilomeli/bwv/layout/genesys/lnf
Netlist library path:/p/acd/proj/work/ade/ilomeli/bwv/layout/netlist/mkskip

x----------------------------------------------------x
NAME devices W(Wmin) Fingers
x----------------------------------------------------x
stack15     2     2     29    The total of transistors is not divisible in the number of rows
stack16     2     2     31    The total of transistors is not divisible in the number of rows
stack17     2     2     33    The total of transistors is not divisible in the number of rows
stack18     2     2     35    The total of transistors is not divisible in the number of rows
stack20     2     2     39    The total of transistors is not divisible in the number of rows
stack21     2     2     41    The total of transistors is not divisible in the number of rows

stack31     3     2     19    The total of transistors is not divisible in the number of rows
stack32     3     2     21    The total of transistors is not divisible in the number of rows
stack33     3     2     23    The total of transistors is not divisible in the number of rows
stack34     3     2     25    The total of transistors is not divisible in the number of rows
stack35     3     2     27    The total of transistors is not divisible in the number of rows


stack43     4     2     15    The total of transistors is not divisible in the number of rows
stack45     4     2     19    The total of transistors is not divisible in the number of rows
stack46     4     2     21    The total of transistors is not divisible in the number of rows

stack54     5     2     15    The total of transistors is not divisible in the number of rows
stack55     5     2     17    The total of transistors is not divisible in the number of rows

stack74     3     4     9     The total of transistors is not divisible in the number of rows
stack75     3     4     11    The total of transistors is not divisible in the number of rows
stack76     3     4     13    The total of transistors is not divisible in the number of rows
stack77     3     4     15    The total of transistors is not divisible in the number of rows
```

## C.1 List of non-valid layout for array of stacked devices structure

| | | | | |
|---|---|---|---|---|
| stack78 | 3 | 4 | 17 | The total of transistors is not divisible in the number of rows |
| stack110 | 3 | 6 | 9 | The total of transistors is not divisible in the number of rows |
| stack111 | 3 | 6 | 11 | The total of transistors is not divisible in the number of rows |
| stack112 | 3 | 6 | 13 | The total of transistors is not divisible in the number of rows |
| stack113 | 3 | 6 | 15 | The total of transistors is not divisible in the number of rows |
| stack114 | 3 | 6 | 17 | The total of transistors is not divisible in the number of rows |

## C.2 List of non-valid layout for the differential pair structure

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x                                              x
x            LIST OF NON-VALID LAYOUTS         x
x                                              x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x Confidential    x
x-----------------------------------------------------x
x This file contains a list of non-valid layouts x
x                                              x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


Today's date = Wed Sep. 6 11:49:28
Machine = plxc25959

Topology: Differential Pair.

Layout Library path: /p/acd/proj/work/ade/ilomeli/bwv/layout/genesys/lnf
Netlist library path:/p/acd/proj/work/ade/ilomeli/bwv/layout/netlist/mkskip

x-----------------------------------------------------x
NAME   W(Wmin) Fingers Rows
x-----------------------------------------------------x
diff15        1      30     3     The total of transistors is not divisible in the number of rows
diff16        1      32     3     The total of transistors is not divisible in the number of rows
diff17        1      34     3     The total of transistors is not divisible in the number of rows
diff18        1      36     3     The total of transistors is not divisible in the number of rows
diff19        1      38     3     The total of transistors is not divisible in the number of rows
diff20        1      40     3     The total of transistors is not divisible in the number of rows
diff21        1      42     3     The total of transistors is not divisible in the number of rows

x-----------------------------------------------------x
```

# Bibliography

[Balasa -99] 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.

[Canaris-91] 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.

[Casier-11] H. Casier, M. Steyaert, and A. H. van. Roermund, *Analog Circuit Design*, Netherland: Springer Netherland, 2011.

[[Chávez-Hurtado-09] L. Chávez-Hurtado, E. Martínez-Guerrero and J. E. Rayas-Sánchez, “Design of reusable CMOS OTAs using CAD tools,” in *IEEE Int. Midwest Symp. Circuits and Systems (MWSCAS 2009)*, Cancun, Mexico, Aug. 2009, pp. 228-231.

[DoITPoMS-00] Cambridge-Dissemination of IT for the Promotion of Materials Science (DoITPoMS), University of Cambridge (2000) E*lectromigration damage* [Online]. Available: https://www.doitpoms.ac.uk/tlplib/electromigration/damage.php.

[Ender-09] Y. Ender and D. Günhan, “Analog layout generator for CMOS circuits,” *IEEE Transactions Computer-Aided Design Integrated Circuits Systems*, vol. 28, no. 1, pp. 32-45, Jan 2009.

Graeb-09] 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 *Design, Automation and Test in Europe*, Belgium, March 2009, pp. 274-279.

[Graeb-11] H. Graeb, F. Balasa, R. Castro-Lopez and Y. Chang, *Analog Layout Synthesis*, Cedar City, UT: Springer, 2011.

[H.-Wu-12] P. H. Wu,. P. H. Lin, T. C. Chen, C. F. Yeh, X. Li and T. Y. Ho, “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.

[H.-Wu-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,” in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 199-212, Feb. 2015.

[Jangkrajarng-03] N. Jangkrajarng, S. Bhattacharya, and R. Hartono, “IPRAIL: intellectual property reuse-based analog IC layout automation,” *Integration, the VLSI Journal*, vol. 36, pp. 237-262, Nov. 2003.

[Kong,-16] D. Kong, D. Seo, and S. M. Lee, “Analysis and reduction of nonidealities in stacked-transistor current sources,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1-5, Aug. 2016.

[L. Geiger-90 ] R. L. Geiger, P. E. Allen, and N. R. Strader, *VLSI Design Techniques for Analog and Digital Circuits*. New York: Mc-Graw Hill, 1990.

[Lewyn-09] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin, “Analog circuit design in nanoscale CMOS technologies,” *Proceedings of the IEEE*, vol. 97, pp. 1687-1714, Oct. 2009.

## BIBLIOGRAPHY

[Lihong-04] Z. Lihong. U. Kleine, “A novel analog layout synthesys tool”, in *Proc. ISCAS* 2004, Vancouver, BC, Canada, May 2004, vol. 5, pp. 101-104.

[Lourenco-06] N. Lourenco, M. Vianello, J. Guilherme, and N. Horta, “LAYGEN—automatic layout generation of analog ICs from hierarchical template descriptions,” 2006 *Ph.D. Research in Microelectronics and Electronics*, Otranto, Italy, Sep. 2006, pp. 213-216.

[Lomelí-Illescas-13] I. Lomelí-Illescas and S. A. Solis-Bustos, “Analysis of the challenges in layout of analog integrated circuits and in the implementation of digital PLLS,” Internal Report *PhDEngScITESO-13-03-R*, ITESO, Tlaquepaque, Mexico, Dec. 2013.

[Lomelí-Illescas-14] I. Lomelí-Illescas and S. A. Solis-Bustos, “Analysis of layout challenges for subnanometric technologies,” Internal Report *PhDEngScITESO-14-16-R*, ITESO, Tlaquepaque, Mexico, Dec. 2014.

[Lomelí-Illescas-16] 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.

[Lomelí-Illescas-16a] I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Conceptual definition of a CAD tool for the automatic layout generation of common analog structures,” Internal Report *PhDEngScITESO-16-02-R*, ITESO, Tlaquepaque, Mexico, Jan. 2016.

[Lomelí-Illescas-17] I. Lomelí-Illescas, S. A. Solis-Bustos, and J. E. Rayas-Sánchez, “Analysis of the implications of stacked devices in nanoscale technologies for analog applications,” in *IEEE Latin American Test Symp. (LATS-2017),* Bogota, Colombia, Mar. 2017, pp. 1-4.

[Maricau-13] Maricau, Elie, Gielen, Georges, *Analog IC Reliability in Nanometer CMOS*, New York City: Springer-Verlag, 2013.

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

[Mohamed-17] 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.

[Mehranfar-90] S. W. Mehranfar, “STAT: a schematic to artwork translator for custom analog cells,” in *Proc. 1990 IEEE Custom Integrated Circuits Conf.*, Boston, MA, pp. 1-4, May. 1990.

[Narendra-90] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan, “Scaling of stack effect and its application for leakage reduction,” in *Int. Symp. On Low Power Electronics and Design*, Huntington Beach, CA, Aug. 2001, pp. 195-200.

[Q. Ma-11] 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.

[Ravindranath-98] N. Ravindranath and F. Terri, “CMOS analog circuit stack generation with matching constraints,” in I*EEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, USA, Nov. 1998.

[Razavi-01] B. Razavi, *Design of Analog CMOS Integrated Circuits*. Boston, MA, McGraw-Hill, 2001.

[Saari-16] 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.

[Saravanan-11] B. Saravanan, T. M. Mohd, and C. C. Heng, “Analog layout design optimization and verification,” in *IEEE Regional Symposium on Micro and Nanoelectronics (RSM)*, Kota Kinabalu, Sep. 2011, pp. 128-131.

[Saxena-13] N. Saxena and S. Soni, “Leakage current reduction in CMOS circuits using stacking effect,” *Int. Journal of Application or Innovation in Engineering & Management (IJAIEM*), vol. 2, no. 11, pp. 213-216, Nov. 2013.

[Shin-16] S. H. Shin, S. -H. Kim, S. Kim, H. Wu, P. D. Ye, and M. A. Alam, “Substrate and layout engineering to suppress self-heating in floating body transistors,” *2016 IEEE International Electron Devices Meeting (IEDM*), San Francisco, CA, Dec. 2016, pp. 15.7.1-15.7.4.

[Subramaniam-12] A. R. Subramaniam, R. Singha, W. Chi-Chao, and Y. Cao, “Leakage reduction through optimization of regular layout parameters,” *Microelectronics Journal*, vol. 43, pp. 25-33, Jan. 2012.

[Unutulmaz -10] A. Unutulmaz, G. Dundar, and F. V. Fernandez, “LDS: a description script for layout templates,” in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, Linkoping, Sweden, Aug. 2011, pp. 857-860.

[Yilmza-09] E. Yilmaz and G. Dundar, “Analog layout generator for CMOS circuits,” IEEE *rans. Computer-Aided Design Integr. Circuits Syst*., vol. 28, no. 1, pp. 32-45, Jan. 2009.

[Zhang-10] L. Zhang and Z. Liu, “A performance-constrained template-based layout retargeting algorithm for analog integrated circuits,” in *Proc. 2010 Asia & South Pacific Design Automation Conf. (ASPDAC '10)*, Chiba/Tokyo, Japan, Jan. 2010, pp. 293-298.

# Author Index

# Subject Index