Faculty of Engineering
Cairo University

AUTOMATED CURRENT MIRROR LAYOUT TOOL (ACML)

Prepared by:

Abanoub Atef Farid

Marina Hamdy Zaky

Mario Adel Messiha

Nada Khaled Ahmed

Omar Mohamed Ezzat

Omar Tarek Abdel-Wahab

Under the Supervision of Associate Prof. Hassan Mostafa

A Graduation Project Report Submitted to

The Faculty of Engineering at Cairo University in

Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in

Electronics and Communications Engineering Faculty of Engineering, Cairo

University Giza, Egypt

July 2019

Abstract

Analog circuit design automation is likely to play a key role in the design process of the next generation of mixed-signal integrated circuits (ICs) and application specified integrated circuits (ASICs).

In the digital domain, computer-aided design (CAD) tools are fairly well developed and commercially available to the design community. Unfortunately, the story is quite different on the analog side.

In this thesis, we summarize the work done over a period of nearly 7 months in collaboration with a promising IC solution corporation, Si-Vision, to produce analog physical design tool for the design of analog integrated circuits (ICs) is presented. This tool offers great flexibility that allows analog circuit designers to bring their special design knowledge and experiences into the physical design process to create high-quality analog circuit layouts. Different from conventional layout systems that are limited to the optimization of single devices, our layout generation tool attempts to optimize more complex modules, at this level the tool is working with current mirrors and differential pairs, which are two of the main building blocks of any analog circuit. This tool covers the following three major analog physical designs stages. 1) Pattern Generation: Saves physical designer's time and headache by auto-generating the CM or DP optimum matching pattern according to given inputs. 2) Placement: This placement algorithm features a novel genetic placement stage followed by a fast communication scheme with the router. 3) Routing: The router imitates the man made CM and DP approaches, Specification-based constructive routing finally completes the interconnection of the entire layout. Several testing circuits have been applied to demonstrate the design efficiency and the effectiveness of this tool. 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 design time.

Acknowledgements

We are using this opportunity to express our gratitude to everyone who supported us throughout the graduation project. We are thankful for their aspiring guidance and friendly advice.

First, we want to thank our major advisor Dr. Hassan Mustafa for his encouragement and patience, his care about following up each stage in the project and his suggestions to solve some problems we faced during the project work, so we would like to thank him very much for his support and understanding over this year.

We would like to express our gratitude for the team of brilliant engineers at Si-Vision IC Solution Corporation for their help, support and for the valuable training sessions we received throughout the project including but limited to Eng. Shrouk Shafie, Eng. Mostafa Nashaat, Eng. Islam Nashaat, Eng. Moustafa Hussien, Eng Mohammed Shehata.

Moreover, we would like to thank, in particular, Eng. Fady Atef who provided us an opportunity to join their team as intern. We are extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to us. Without his precious support it would not have been possible to complete this tool.

Last but not least, we want to thank our families for their support, tolerance and love during this year especially during the hard times they were always there having faith in what we do. We are grateful to our families, colleagues and friends for always motivating us, without them we wouldn’t have come so far.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ADC	Analog to Digital Converter
AMS	Analog Mixed Signals
ASIC	Application-Specific Integrated Circuit
CAD	Computer-Aided Design
CM	Current Mirrors
DP	Differential Pair
DRC	Design Rules Check
DM	Design Manual
d	Dummy
ERC	Electrical rules check
NMOS	n-Channel MOSFET
NIMP	N-Implant
NW	N-Well
PMOS	p-Channel MOSFET
PIMP	P-Implant

PW	P-Well
LVS	Layout vs. Schematic
LUP	Latch Up
M#	Metal #

Chapter 1
Introduction

Chapter 1.
Introduction

In this thesis, we are going to propose a technique to fully automate the layout of any current mirror with any specification (Width, Length, number of fingers and multipliers ……..)

1.1. Motivation

During the past two decades, the electronics industry has grown very fast both in Size and in complexity. Designers began talking about chip design only 25 years Ago. At the beginning, the idea was to design chips to reduce the computer size.
Instead of room-sized computers, we have now ended up with PCs running at a Speed that back then was considered “impossible to imagine.” The application of IC technology has exploded into many parts of our lives.

IC layout design was originally hand-drafted on special paper called Mylar.
This was a long and laborious task. The market demands and advances in technology brought about an immediate need to develop software and hardware
Solutions to improve the time-to-market of the chip designs and especially to automate the entire process. Accuracy of the final masks was also a driving force in the computerization of layout design.

The first platforms were custom built to ensure that graphics applications ran quickly and had sufficient capabilities. Companies such as CALMA (Data General) built mainframe-sized machines and developed specialized software for printed circuit board (PCB) and integrated circuit (IC) applications.

The disk size was huge by today’s standards. The top-of-the-line computer had 220MB of disk space and only 0.5MB of DRAM was available at the time.
The price tag was around $1 million U.S., and not everybody could afford to be involved in this kind of design. As the market and the chip sizes grew and more companies were involved in chip design, the hardware and software developers came up with faster, smaller, and cheaper solutions.

The biggest revolution in hardware was the development of the “engineering workstation,” which ran a version of the UNIX platform. Workstations have developed over the years to incredible speed and complexity. They are used for all kinds of engineering design, so the prices are very affordable. HP, Sun, and IBM are only a handful of survivors in this field, Daisy being one that has disappeared from the market. Today there is tremendous pressure to go to even cheaper and more popular platforms, such as PCs with Linux and Windows NT platforms.
As the hardware platforms evolved, software development progressed at an even faster rate. Companies such as Mentor Graphics, Cadence, Compass, and Daisy gained larger and larger shares of the IC and PCB design tools market. For the PC platform, a company such as Tanner, with a product called L-Edit, is an example of how the software development market has grown for IC design.

The direction for development of the software has really been toward more and more automation of the tasks that are labor intensive: for example, designs with hundreds of transistor blocks, where interconnection analysis is impossible to do by human eyes, or verification of a 256-MB memory chip.

Significant examples of automation include the following:

***Layout synthesis**:* Layout can be created from “code” instead of the traditional methods of manually drawing the polygons.
***Layout migration**:* Alternatively, layout can be “migrated” from one set of design rules to another using mapping and sophisticated compaction techniques.
***Layout verification**:* These tools perform an increasing number of checks on the final layout before it goes to production. For example, minimum size rules are checked to ensure that the design is manufactural.
***Circuit synthesis**:* Similar to layout synthesis, in this case schematics can be automatically generated from specialized “code” (i.e., VHDL or Verilog).
This has had a huge impact on layout design, as the sheer volume of circuitry produced by these circuit synthesis tools created a need for more layout automation such as place-and-route tools.
***Place-and-route**:* Instance placement for literally millions of cells as well as optimizing the placement for minimum connectivity and maximum circuit performance.

Today, layout design is carried out in an environment that is ever changing.
The software tools and approaches, computing platforms, the companies providing these tools, the customers we serve, the applications that are being implemented, and the market pressures we face are all changing year by year.

These changes make this industry an interesting one in which to be involved.
However, let’s not forget that the fundamental concepts behind producing quality layout are based on physical and electrical properties that never change.

1.2. Analog Layout Generation and Verification processes

The following figure shows a general layout design flow that is applicable for all design types. This procedure is straightforward and self-explanatory and could be applied to almost any engineering task. Of course we will concentrate on how it applies to layout design.

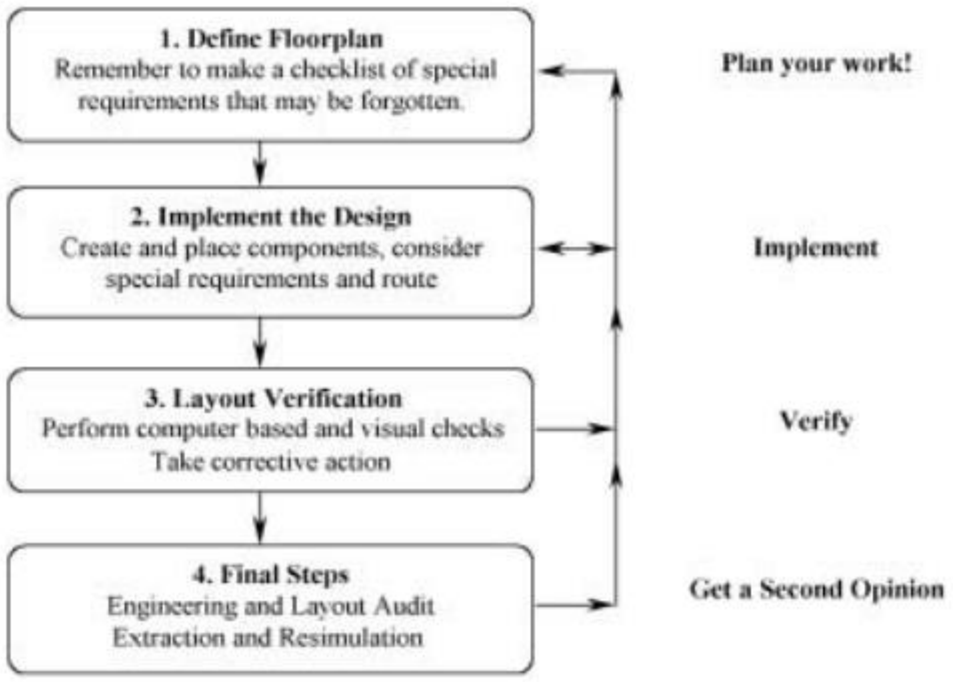

1. Define Floorplan
Remember to make a checklist of special requirements that may be forgotten.
2. Implement the Design
Create and place components, consider special requirements and route
3. Layout Verification
Perform computer based and visual checks
Take corrective action
4. Final Steps
Engineering and Layout Audit
Extraction and Resimulation
Plan your work!
Implement
Verify
Get a Second Opinion

Figure 1: Layout design procedure

Step 1 is crucial for getting started on the right track. It is in this step that we collect and review our knowledge of layout design and apply it to the specific circuit design under consideration. The aim is to produce a strategy for attacking the design by documenting the general areas where all components and signals will go.

Step 2 is simply implementing the design: executing and possibly revising the floorplan based on the realities of implementation. One way to think of the design process is to “plan top down,” then “implement bottom up.” By this we mean that we first floorplan general areas and approaches with a top-level view. With this plan in place we implement the design by starting with the lowest level components first and fill the areas defined in the plan. The lower level design tasks are easier because the constraints imposed on them were defined in the top-level floorplan. As the general areas are completed, we adjust our plan for future work. With a sound floorplan, adjustments are minor and the completion of the design is easy.

Computer-based checks (DRC and LVS) form the bulk of step 3. These checks should be done in a certain order.

On top of the computer-based checks a visual inspection is recommended, as the automatic computer checks are only as good as the rules that are coded into them. Make a plot of your design and look at it. Also, there are many aspects of most designs that cannot be checked by computer. An example of this is the degree of symmetry of a balanced layout. These visual checks should be part of the audit checklist as a reminder.

Step 4 is a final sanity and cross-check to confirm that all requirements have been met and none missed, along with a final extraction step.

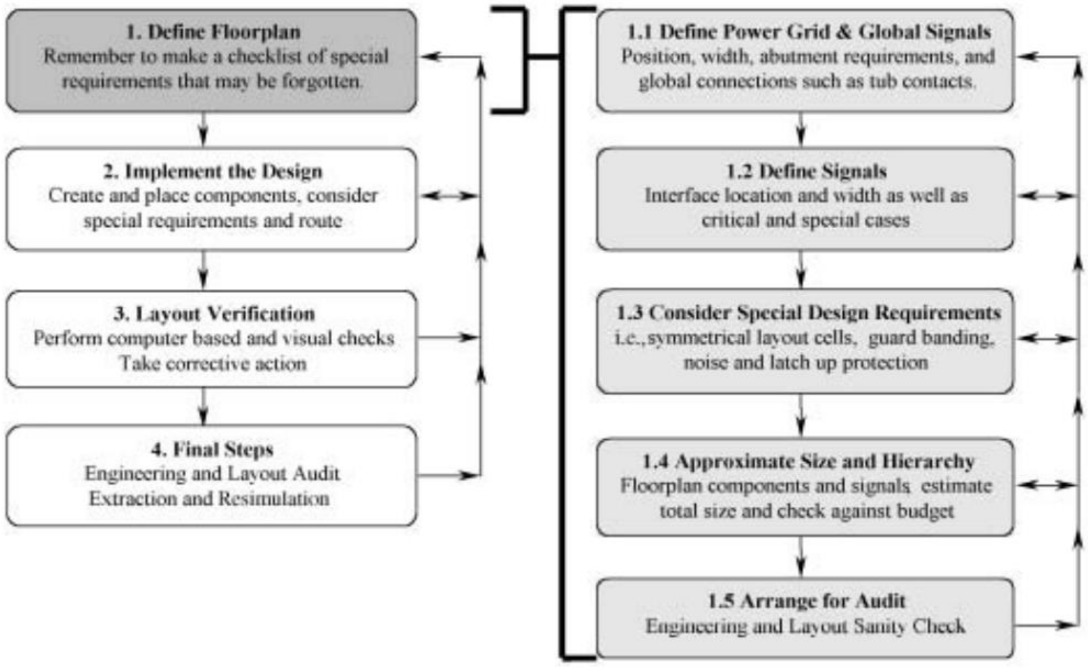

1. Define Floorplan
Remember to make a checklist of special requirements that may be forgotten.
2. Implement the Design
Create and place components, consider special requirements and route
3. Layout Verification
Perform computer based and visual checks
Take corrective action
4. Final Steps
Engineering and Layout Audit
Extraction and Resimulation
1.1 Define Power Grid & Global Signals
Position, width, abutment requirements, and global connections such as tub contacts.
1.2 Define Signals
Interface location and width as well as critical and special cases
1.3 Consider Special Design Requirements
i.e., symmetrical layout cells, guard banding, noise and latch up protection
1.4 Approximate Size and Hierarchy
Floorplan components and signals, estimate total size and check against budget
1.5 Arrange for Audit
Engineering and Layout Sanity Check

Figure 2: Steps of floor planning

The first step, 1.1, is related to the planning of the layout of the power supplies and/or global signals. The power supply connectivity is typically called the power grid. Power supply resistance from the interface to all parts of the design must be considered. In this case special attention must be applied to the width of the supply lines and the grid or mesh of power lines through the design. Again the interface to other designs is important, especially in the case of a cell design where it may be desired to array it or have seamless abutment requirements to other cells. Let’s not forget that tub and substrate contacts are typically connected from the power supplies, so a strategy to lay out these contacts must also be considered. Step 1.2 is to list all of the input and output signals. Each signal is assigned a position on the interface of the design to the neighboring designs. The interface is defined as the boundary of the design. In some cases certain signals will have a specific or nondefault signal width assigned to them. Special considerations for signals may include clock signals, signal buses with multiple bits that need to be matched between them, critical path signals, and shielded signals. In Step 1.3 we have to deal with special design requirements such as layout symmetry, specific requirements for latch-up protection, or noise immunity. More examples of special design requirements might be that the design must be pitch matched (i.e., limited in size in one direction), must have a very specific critical path signal, or might be a nonstandard part of the design. Step 1.4 is very important to help finalize the size of the design and estimate the feasibility of meeting all of the design requirements within the area and schedule constraints. Using any previous knowledge about older designs of the same complexity and the process design rules, a layout designer can approximate the size of each component and the complete design. The number of different components to be implemented can be identified and the overall hierarchy or partitioning of the design can be completed. Areas for internal routing and signal connections should be allocated. The routing layer in each interconnect area should be identified. Extra signals or space should be reserved since we have only an educated estimate as to the size of the final design. At this point we should have a preliminary floorplan and implementation strategy. The floorplan should comprise a definition of the interface or boundary with all of the signal ports assigned to their proper locations. Signals with special requirements are identified, and the area impact of these special signals is included in the total area estimate. If it is a hierarchical design, subcomponents are also known with their respective interfaces defined. Spare space and spare signal lines are included in this plan. Step 1.5 is a sanity and cross-check to confirm that all requirements have been met and none missed. There are requirements related specifically to layout guidelines and styles for the process, but also circuit design requirements as well. The floorplan is a communication tool between the layout and circuit designer, as the circuit designer most likely had defined some specific requirements for his or her design and had assumed some kind of layout floorplan in modeling the design for its environment. It is very important to involve the circuit designer in the audit process. The layout audit also relates to the next level of integration for the piece that you designed. The “brick” that you have now floorplanned has to interface perfectly to all of its neighbors and their interfaces; otherwise, when put together it won’t work exactly as planned. As an example, the top-level chip design has to fit within the context of the chip package, so a review of any floorplan is very important. The person responsible for integrating your design should audit this floorplan as well to review the requirements related to size of the design, the layout architecture or approach, and your designs interface, among other things. It is not uncommon for audits of really complicated floorplans to require two people: one to check the engineering requirements and another to check the layout needs. The auditor(s) should be a person who is not directly involved in the design, but who has the expertise to evaluate and appreciate the floorplan quality, and to make constructive comments. To help the auditor perform a proper audit, ideally a checklist is used for each type of layout design. Refer to the addendum checklists for some examples.

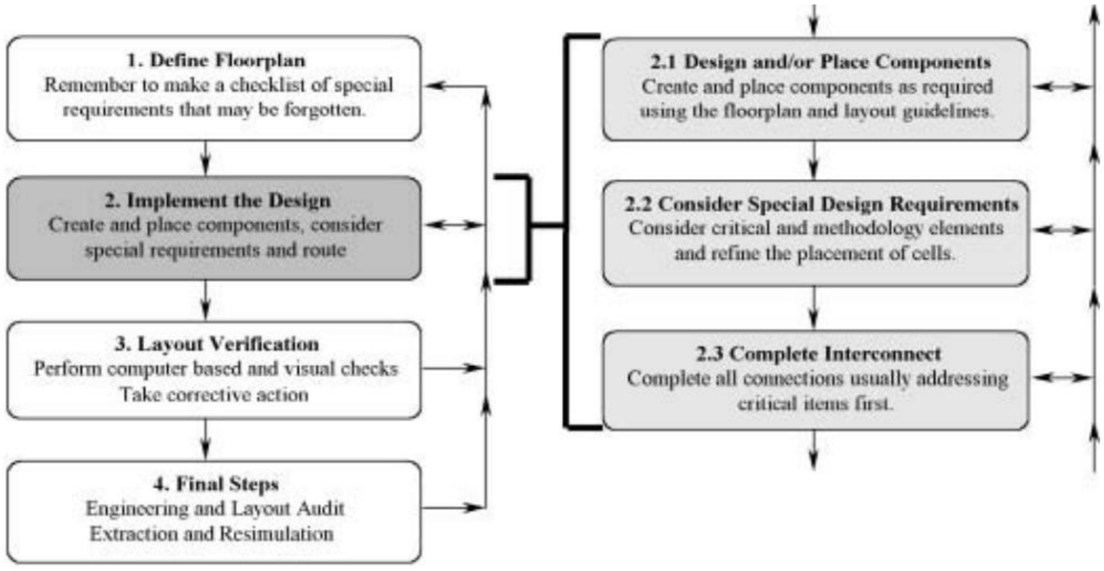

1. Define Floorplan
Remember to make a checklist of special requirements that may be forgotten.
2. Implement the Design
Create and place components, consider special requirements and route
3. Layout Verification
Perform computer based and visual checks
Take corrective action
4. Final Steps
Engineering and Layout Audit
Extraction and Resimulation
2.1 Design and/or Place Components
Create and place components as required using the floorplan and layout guidelines.
2.2 Consider Special Design Requirements
Consider critical and methodology elements and refine the placement of cells.
2.3 Complete Interconnect
Complete all connections usually addressing critical items first.

Figure 3: Design implementation steps

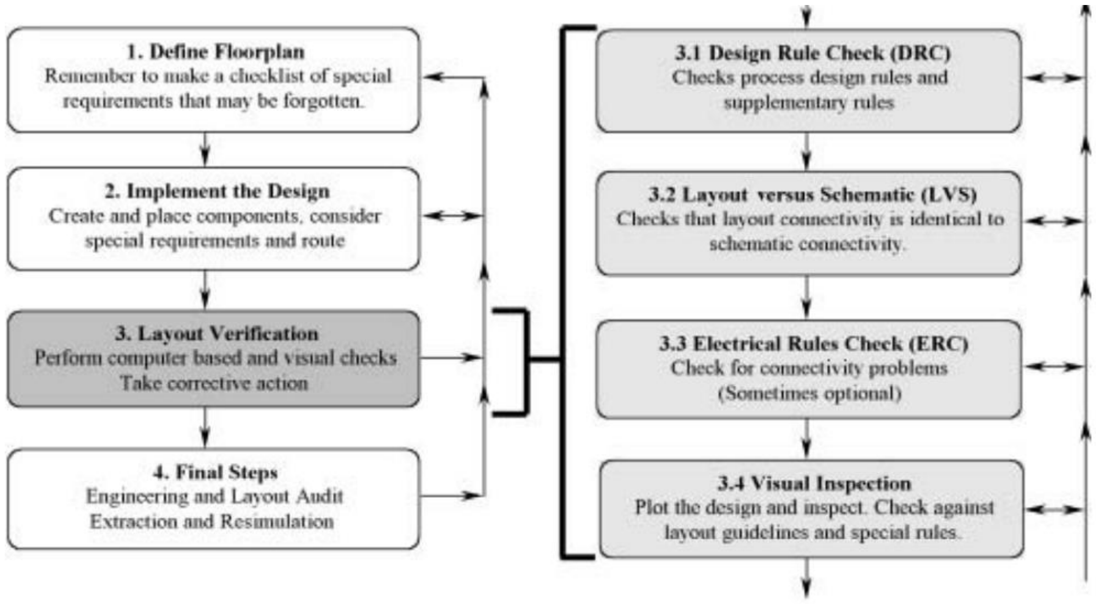

1. Define Floorplan
Remember to make a checklist of special requirements that may be forgotten.
2. Implement the Design
Create and place components, consider special requirements and route
3. Layout Verification
Perform computer based and visual checks
Take corrective action
4. Final Steps
Engineering and Layout Audit
Extraction and Resimulation
3.1 Design Rule Check (DRC)
Checks process design rules and supplementary rules
3.2 Layout versus Schematic (LVS)
Checks that layout connectivity is identical to schematic connectivity.
3.3 Electrical Rules Check (ERC)
Check for connectivity problems
(Sometimes optional)
3.4 Visual Inspection
Plot the design and inspect. Check against layout guidelines and special rules.

Figure 4: Layout verification steps

- Design Rule Check (DRC): The design rule verification step checks that all polygons and layers from the layout database meet all of the manufacturing process rules. These design rules define the limits of a manufactural design. Width and space rules fall into this category. Meeting the manufacturing requirements is the absolute minimum rule set that must be checked and corrected. Because this is the first level of verification once the layout is implemented, typically many methodology, connectivity, and guideline rules are checked as well. We refer to these as a set of supplementary rules. An example would be an illegal use of layers (ESD layer in the logic area) or illegal devices or connections. Tip: A truly complete DRC verification approach would be to verify not only the design that you, as a designer, have implemented, but also your design placed within the context in which it is going to be used. If the specific components that will interface or be adjacent to your design are available, perform a DRC check with this interface cell included. If your cell is a general-purpose design, then a more intricate and exhaustive check should be performed, perhaps including all possible interface cells as well as different
- Layout versus Schematic (LVS): LVS verification is checking that the design is connected correctly. The schematic is the reference circuit and the layout is checked against it. In principle, the following is verified:
  • Electrical connectivity of all signals, including input, output, and power signals to their corresponding devices.
  • Device sizes: transistor width and length, resistor sizes, capacitor sizes.
  • Identification of extra components and signals that have not been included in the schematic; floating nodes would be an example of this.
  The last item overlaps into the items checked in the electrical rules check, which is described next.
- Electrical Rules Check (ERC): As noted in Figure 4 the ERC is sometimes an optional or seldom used as an independent verification step. Many of the issues are caught by the LVS check, and thus the ERC has become redundant. Electrical rules checked in this step are usually limited to errors in connectivity or device connection. Examples include the following:
- • Unconnected, partly connected, or extra devices.
  • Disabled transistors.
  • Floating nodes.
  • Short circuits.
  • Special checks not checked elsewhere (i.e., antenna rules).
  As a subset of the LVS check, an ERC generally executes more quickly and therefore is useful to accelerate debugging problems such as a VDD-to-VSS short circuit.

In the next section, and after giving a complete view of what an analog Layout design flow is, we are going to state the problems of the Layout process, that makes the need of automation mandatory.

1.3. Problem statement

Unlike the digital domain, where Computer-Aided Design tools (CAD tools) are fairly well developed and commercially available to the design community, and where automatic place and route (APR) ae used to perform the layout with significant sped up in the placing and routing processes, the story is quite different on the analog side.

In fact, physical design automation of analog IC design has seen significant improvement over the past decade, however, it has reached the rate of its digital counterpart. This shortfall is primarily rooted in the analog IC design problem itself, which is very much more complicated even for small problem sizes.

Although analog modules typically contain only a small number of devices compared to digital ones. Analog designs are characterized by a much richer and more complex set of design constraints that need to be considered simultaneously and which may span several domains (e.g., electrical, electro-thermal, electro-mechanical, technological, geometrical domain), this makes the effort needed to automate the analog physical design much more harder than the one needed to automate the digital flow.

Due to the special and necessary constraints imposed on analog layouts, such as:

- Large variation of MOS transistor sizes.
- High Sensitivity to parasitic capacitance. (Ignoring parasitics can kill the chip),

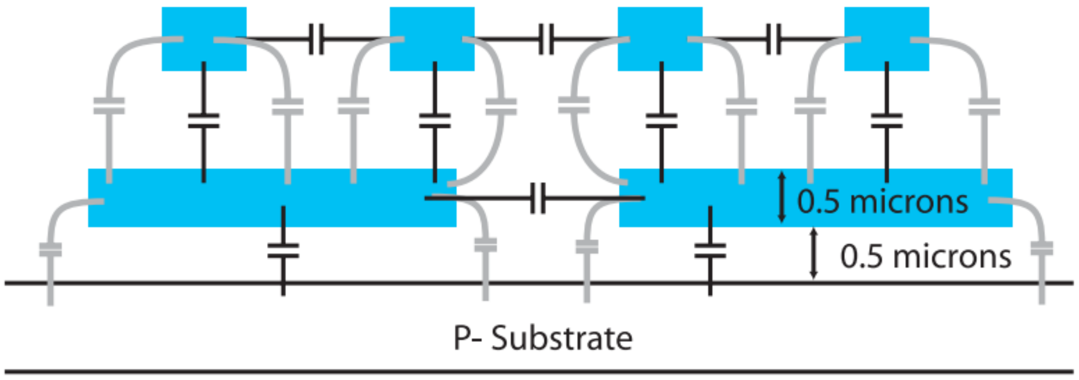


Figure 5: Capacitance is everywhere. Everything is talking to everything else.

- Crosstalk:

- Special device matching requirements for some circuits. (e.g. current mirrors, differential pairs… etc.)
- Symmetry requirements. (Essential to ease the flow of electrons, if the electron life is easy, the chip will perform properly).
- IR drops (In many cases, routing using the minimum width defined in the PDK may degrade the performance from the point of view of the voltage drop, in this case, thicker wires are required, this must be put into consideration).
- Current capability and electro migration.
- Temperature gradients.

The previous constraints maybe divided into four main categories:

(1) Technological constraints: that enable the fabrication and are derived from technology used.

(2) Electrical constraints: that ensure the desired electrical behavior of the design.

(3) Geometric constraints: that are introduced to reduce the overall complexity of the design process, and enhances the yield.

(4) Commercial constraints that arise from chip area or packaging requirements.

This constraint-related problem also makes algorithm and tool development for analog IC design much more difficult as the number of specific design algorithms needed may increase with each new constraint type. Considering today's conventional design algorithm development approach (one constraint type and one algorithm to handle it), its weakness becomes all too apparent when faced with new complex constraints affecting multiple design parameters simultaneously and vastly outnumbering simple constraints. This is one of the primary reasons why conventional analog design automation still lags behind its digital counterpart.

In this project's thesis, our driving constraints are:

1) Area: as we are making the layout of bias circuits, which are usually given a small area to fit in the floorplan.

2) Placement and routing matching: to ensure that all devices observe the same effect, and are etched in the same manner, and other issues that will be declared

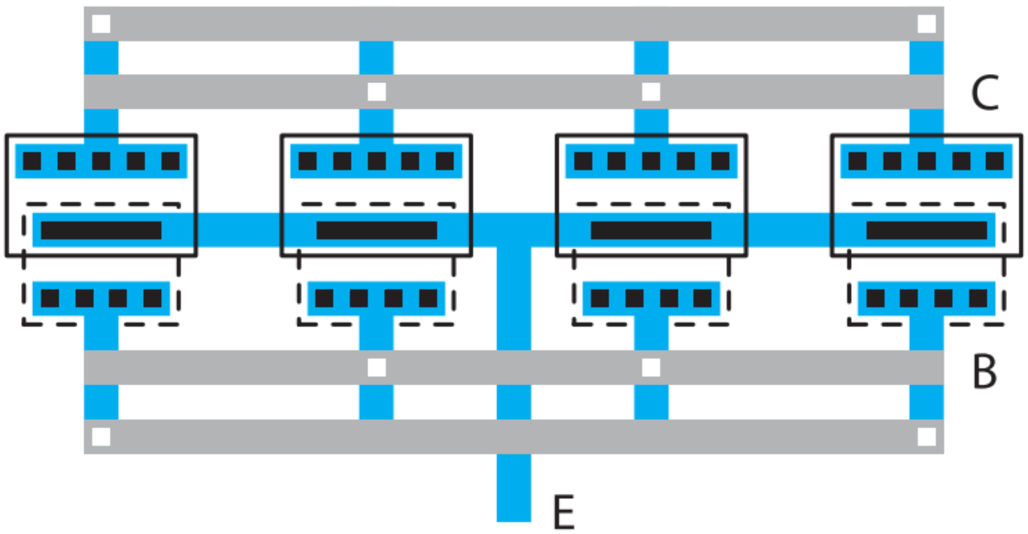


Figure 6: An example for well-matched differential pair

in details in the fourth chapter.

3) Current capability and electro migration consideration.

4) Symmetry constraint:

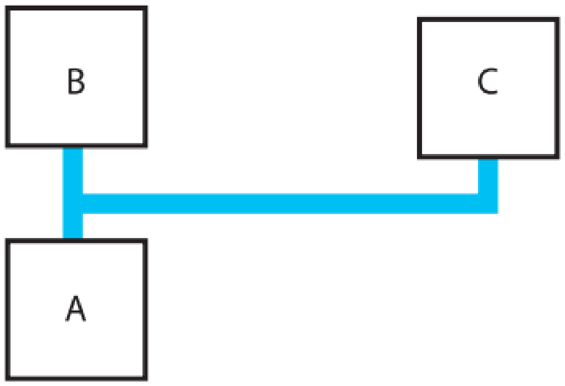


Figure 7: Casually placed layout, poor matching.

Symmetry is a major concern in matching devices.
In Figure 3, the output from block A is correctly wired to blocks B and C. It's a fair layout. It will work.

However, block C has a much bigger parasitic on it because you have given it a longer wire. A nicer way to wire these blocks is to imagine a line of symmetry between the blocks, then place the blocks in mirror image on either side of the

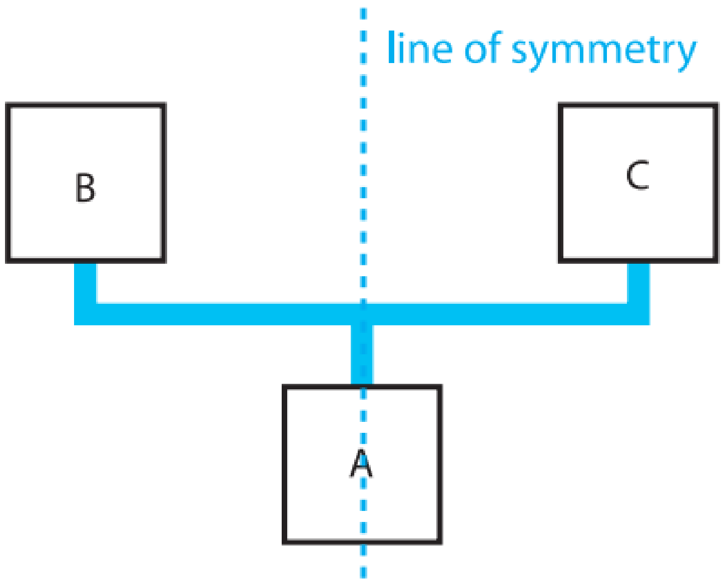


Figure 8: Imaginary line of symmetry helps components match.

line.

5) Parasitic capacitances.

These solution approach for all these complex constraints will be described in the coming section.

1.4. Solution approach

This tool tries to use all the constraints mentioned above and combine them to utilize the best solution for the layout (placement and routing).

The tool first takes the total number of devices & minimum spacing of devices from the previously defined PDK and uses these information to determine the best matching pattern for the desired block and then the placement of this patter, taking into consideration all the rules required for perfect matching, such as:

- Keeping devices in the same orientation.
- Placing the root component in the middle.

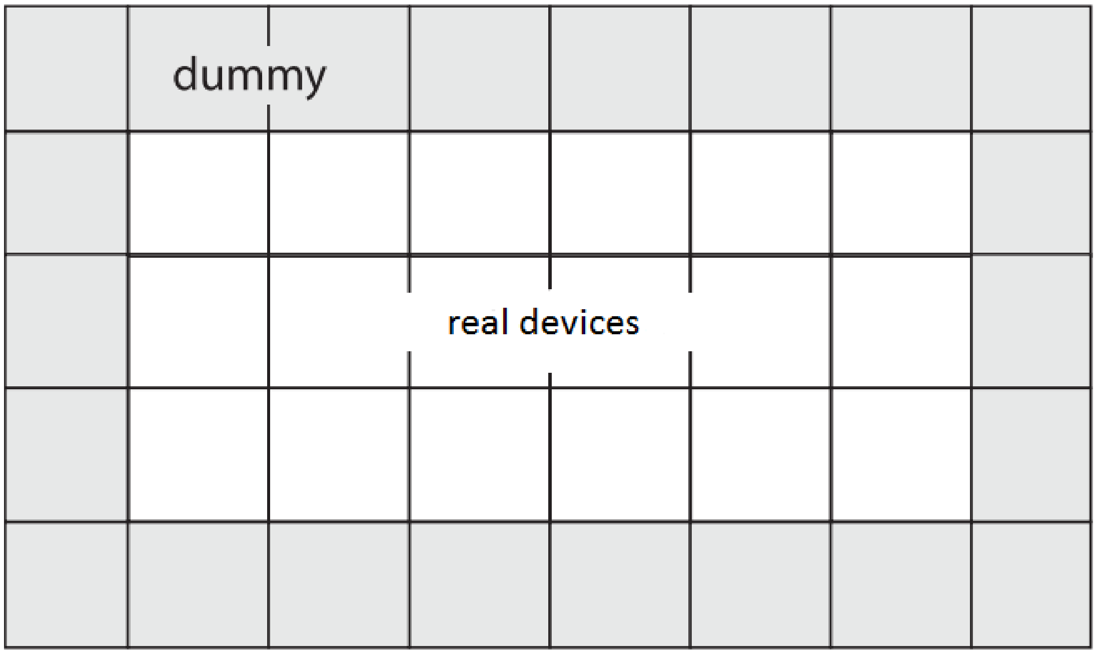


Figure 9: Real devices encased within a ring of dummy devices.

- Surrounding the devices with dummies to provide protection:
- Match the parasitics on wirings.

- Keeping everything in symmetry.

After achieving the best matching and placement requirements then comes routing phase. While routing the tool trying to preserve some very important constraints first the current capability of each route (make sure that each route can carry the current flowing through it over long periods of time without suffering electro-migration problems)

At the same time the tool is trying to keep the routing of devices a symmetric as possible so that each device can experience same effects from the point of view of parasitics.

The last concern the tool takes in consideration to keep the routing area as minimum as possible by making use of the area above the devices as the tool works for base band circuits that are not very sensitive to parasitics effects .

1.5. Organization

In the following chapters we will describe in details the work done and the final output of this work.

In chapter two there is the previous work done in this area and the differences between it and ours.

In chapter 3 we will give an overview of the tool, how to use it and what is the outputs and options available.

In chapter 4 we will talk about different matching techniques and explain in details our algorithm for common centroid matching.

In chapter 5 we will explain in details the routing technique we used and how it was done.

In chapter 6 we will show the results from the tool with explanation for how it works.

In chapter 7 there is a brief summary for this thesis and the future work that will be done.

Chapter 2.
Background and Related work

2.1. Introduction

Historically analog designers were almost mysterious and old-school as radio frequency (RF) designers. After acquiring years of experience manipulating the many parameters not present in digital designs, analog designers become fountains-of-knowledge in how to add the "secret sauce" to all the different sorts of analog circuits in use today,

Digital IC design has been largely automated with high-level languages, RTL coding, logic synthesis, and automated place and route tools. What about analog IC layout automation, is it possible? A few EDA companies think that it is possible and even practical, but they are not sure

The reason is that there are many more circuit types, each with many more parameters to be optimized, than in the typical digital circuit. Advances have been made for specific circuit types in recent years, but much more needs to be done before analog can be said to have caught up with digital EDA. There are two techniques was developed, which have since been transferred to Cadence, namely "continuous design flow" and "bottom-up meets top-down" design flows.

A continuous layout design flow is a blind spot in today's traditional analog flows,

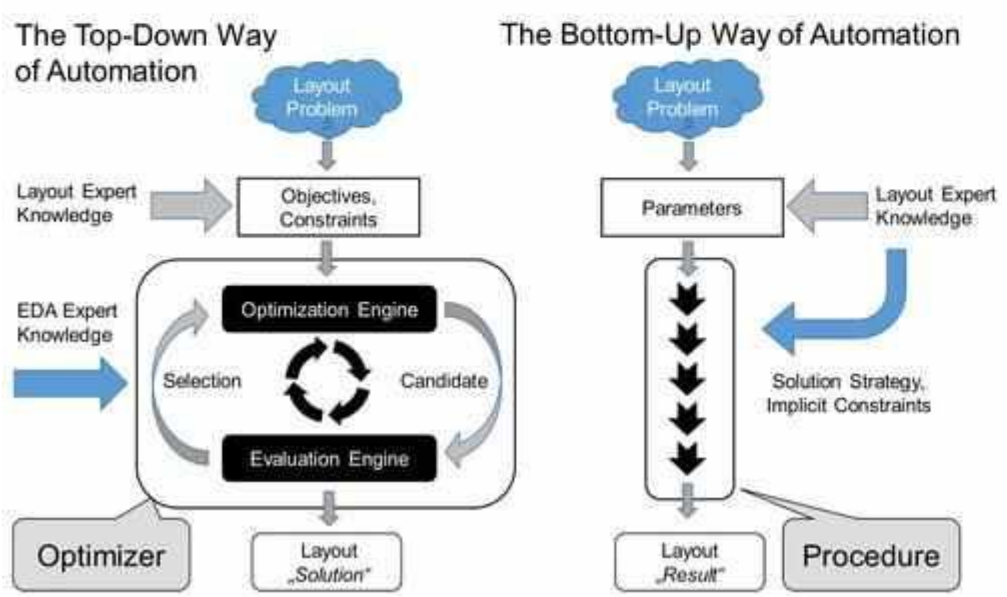


Figure 10: Layout automation solution approaches

The automation of ANALOG circuit design is likely to play a key role in the design process of mixed-signal integrated circuits (ICs) and application specified integrated circuits (ASICs.)

In the digital domain, computer-aided design (CAD) tools are fairly well developed and commercially available to the design community.

Unfortunately, the story is quite different on the analog side.
Today, analog circuits are still largely being handcrafted with only a SPICE-like simulation shell and an interactive layout environment as the supporting facilities. Due to the special and necessary constraints imposed on analog layouts (e.g., large variation of MOS transistor sizes, sensitivity to parasitic capacitance, crosstalk, device matching, symmetry requirements, voltage drops, current density, temperature gradients, piezoelectric effects, electro migration, etc.),

The analog layout design is intrinsically more difficult than its digital counterpart this is due to technology scaling, the device electrical properties worsen in analog circuits and the design window decreases. Electrical parameters caused by different layout topologies have already become necessary for high-performance circuits even at the circuit design stage. As a result, although analog circuits typically occupy only a small fraction of the total area of mixed-signal ICs, their designs have become the bottleneck, both in design effort and time, as well as test cost. Moreover, analog circuits are often responsible for the design errors and the expensive design iterations.

So far, several CAD tools have been developed to automate the generation of analog layouts. However, as some of these tools are developed for the system designers, the circuit designers tend to experience difficulties to integrate their design knowledge and experiences into the design flow. We will demonstrate a novel automated layout design tool, named as automatic layout design aid for analog integrated circuits (ALADIN) also we will tackle many tools called LAYLA and AIDA. This tool (ALADIN) allows analog circuit experts to integrate their specific knowledge and experiences into the synthesis process to create high-quality layouts. The designers can construct layouts made of parameterizable modules in a technology and application-independent fashion. The placement and the routing of modules are performed automatically under the constraints tailored for applications.

these tools used to automate the generation of analog layouts have been developed. Usually, these tools apply a top down approach taking the already optimized circuit netlist as input and generating the layout for the circuit.

The use of procedural generators is one of the most mature layout automation techniques used for analog circuits. In these systems, specific software is written to support each unique circuit topology and the analog specific knowledge about the layout is coded into the software itself. Its major disadvantage is that they require a large coding effort for each new topology. Therefore, it is not a good choice for an environment requiring a large variety of circuit topologies.

Conway and Schrooten two famous scientists developed a template driven analog layout system. For a given module type, a template module layout is created once by an expert designer. To obtain real layout, the modules in the template are enlarged, compacted and then connected. Although this technique produces good quality layout in a reasonable amount of time, a template has to be created for each new type of circuit that limits the generality of this method. Moreover, the library of templates has to be updated for each new technology.

A variety of analog layout generation and some tools are reviewed next. First attempts for analog layout automation was known as procedural generation. Procedural module generation used parametric cell which is commonly known as pCell. The parametric cell concept is coding the entire layout of a circuit in a software tool. Hence, it is fully developed by the designer. A tool called ALSYN uses fast procedure algorithms.

However, this method lacks the flexibility to adapt to wide changes. Hence, the cost of making a new design is high and technology migration may need redesign.

The second type is template based. It uses a pre-defined template that has the relative position and interconnection of devices. A tool called IPRAIL, which stands for Intellectual Property Reuse-based Analog IC Layout, extracts the information from already made layout automatically for retargeting purposes. Layout retargeting means using an existing layout to generate a new layout. Hence, this method can be used for technology migration, update specifications, or optimizing the old design. A tool is introduced by Po-Hsun to generate a new layout by integrating existing design expertise.

The third type is optimization based. These approaches synthesize the layout by means optimization techniques based on some cost functions. This type is easier to implement compared to procedural and template-based approaches.

Now, let us consider commercial solutions. Virtuoso® Layout Suite Family made the creation and navigation in a complex design. This method manages multiple abstraction levels for device, cell, block, and chip levels. Moreover, it contains different level of assistance Basic design creation assisted correct-by-construction wire-editing functionalities to ensure real time process, Hierarchical designs intent for schematic editor. Synopsys'® Galaxy Custom Designer LE also offers a set of automation functionalities for layout generation. It has an automatic guard ring generator which creates guard rings in real-time and it is easily edited. Furthermore, user assisted applications as auto connect features, interactive bus routing with different via pattern choices align assist displays with interactive alignment, commands to specify locations to create bridges or a tunnel for routing, these frameworks adds several functionalities to speed-up layout design.

The gap between electrical and physical design needs to be closed by ensuring that the post layout performance is guaranteed in the presence of layout parasitic. Iterations between the physical design and the electrical simulation is ineffective as it is a time-consuming process. The proposed solution is using a layout-induced effects in the circuit sizing phase to overlap electrical and physical design phases. Hence, parasitic-aware, layout-aware or layout-driven sizing should be made. Device parasitic effects are modelled by linear regression obtained by sampling the design space and with means of procedural generator it produces the layout for each point, where each solution is aware of its specific layout induced effect. Hence, the effects can be expected without actually generating the layout in real-time. Moreover, the parasitic effects are modelled in the circuit equations. Hence, a deterministic nonlinear optimization algorithm is used to determine the device sizes.

2.2. Prior work

1. AIDA tool {Analog Integrated Circuit Design Automation}

The Analog IC Design Automation (AIDA) framework implements an analog IC design flow from circuit-level specification to a physical layout description focusing on design optimizing and porting using highly efficient searching methods combined with accurate circuit-level simulation, layout design rules and parasitic extraction engines.

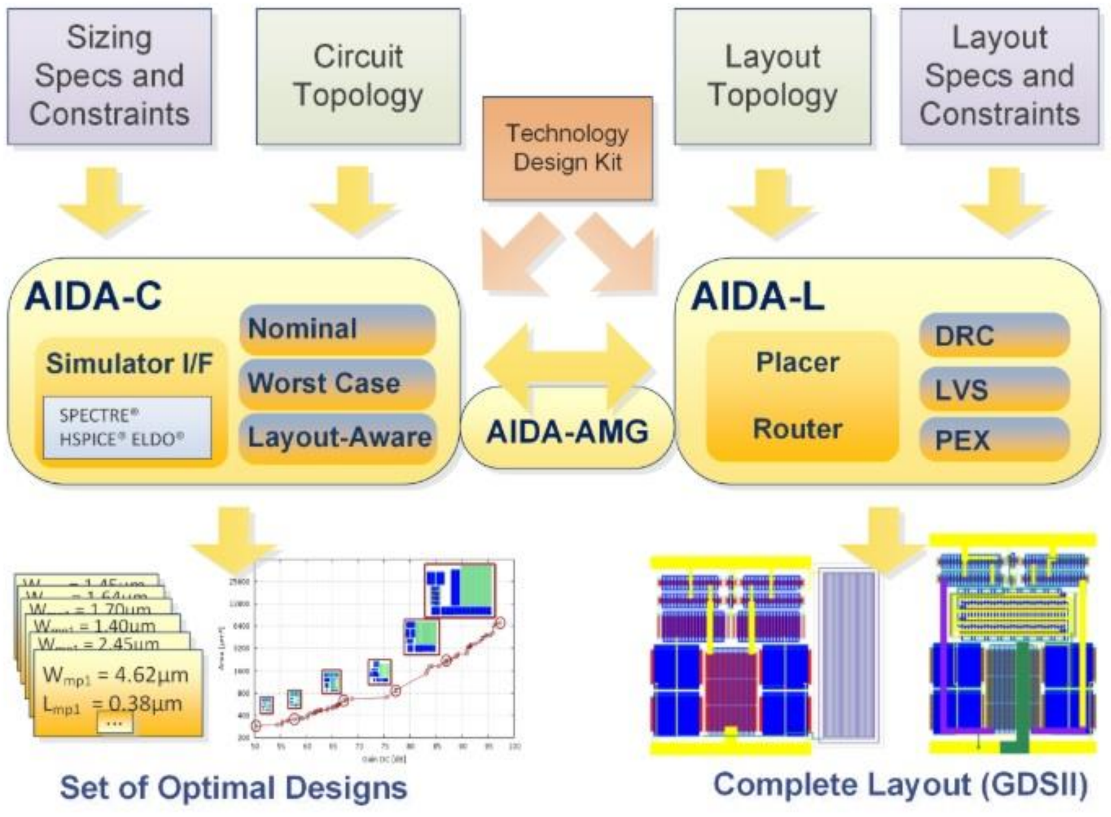


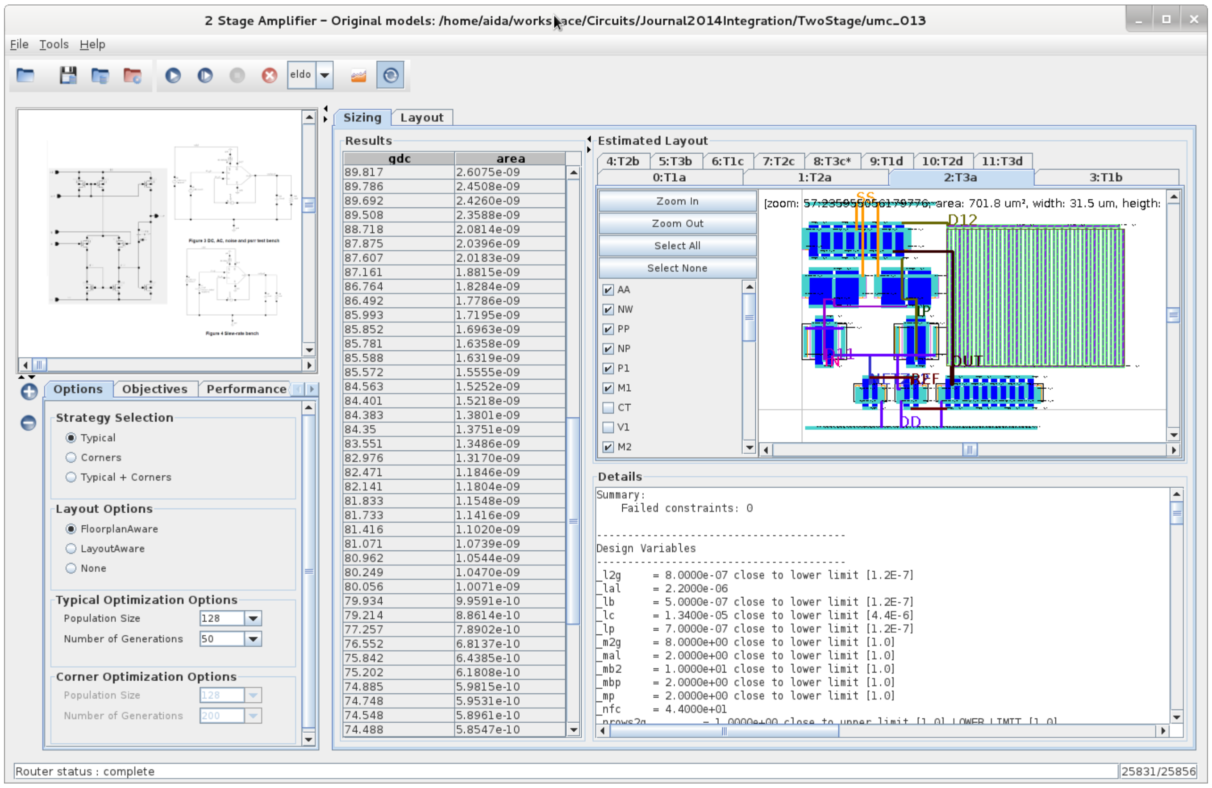
Figure 11: AIDA tool

AIDA assists expert analog IC designers by automating the most time-consuming and repetitive tasks from the typical analog IC design flow. **AIDA-C** automates circuit sizing with high accuracy and within a reduced time frame. **AIDA-L** generates the complete layout for sized circuits, from device placement to detailed routing.

Analog and Mixed-Signal (AMS) systems are found in a wide range of applications, such as, communications, medical or multimedia applications. These extremely competitive markets force the analog designer to face the complex design challenges within strict and demanding timeframes. Today's analog design is supported by circuit simulators, layout editing environments and verification tools, however the design cycle for AMS ICs is still long and error-prone.

AIDA's framework targets the analog IC design automation while keeping the common design flow by involving the designer and his (her) knowledge and focusing on efficiently automating repetitive design tasks. Thus, facilitating design reuse and fast response to specification changes of analog cells like VCO, LNA, voltage mixer, differential amplifiers, band gap voltage references, integrators, comparators, etc..

AIDA framework is composed by two main modules: AIDA-C and AIDA-L. AIDA also includes an intuitive GUI allowing the designer to manage and interact with the design automation process.


2 Stage Amplifier – Original models: /home/aida/workspace/Circuits/Journal2014Integration/TwoStage/umc_013
File Tools Help
eldo
Options Objectives Performance
Sizing Layout
Estimated Layout
4:T2b 5:T3b 6:T1c 7:T2c 8:T3c* 9:T1d 10:T2d 11:T3d
0:T1a 1:T2a 2:T3a 3:T1b
Zoom In
Zoom Out
Select All
Select None
AA NW PP NP P1 M1 CT V1 M2
[zoom: 57.235955056179776, area: 701.8 um², width: 31.5 um, heigth:
SS
D12
IP
IN
OUT
VBIAS
DD

Strategy Selection
- [✓ Typical]
- Corners
- Typical + Corners

Layout Options
- [✓ FloorplanAware]
- LayoutAware
- None

Typical Optimization Options
Population Size 128
Number of Generations 50

Corner Optimization Options
Population Size 128
Number of Generations 200

Results

gdc	area
89.817	2.6075e-09
89.786	2.4508e-09
89.692	2.4260e-09
89.508	2.3588e-09
88.718	2.0814e-09
87.875	2.0396e-09
87.607	2.0183e-09
87.161	1.8815e-09
86.764	1.8284e-09
86.492	1.7786e-09
85.993	1.7195e-09
85.852	1.6963e-09
85.781	1.6358e-09
85.588	1.6319e-09
85.572	1.5555e-09
84.563	1.5252e-09
84.401	1.5218e-09
84.383	1.3801e-09
84.35	1.3751e-09
83.551	1.3486e-09
82.976	1.3170e-09
82.471	1.1846e-09
82.141	1.1804e-09
81.833	1.1548e-09
81.733	1.1416e-09
81.416	1.1020e-09
81.071	1.0739e-09
80.962	1.0544e-09
80.249	1.0470e-09
80.056	1.0071e-09
79.934	9.9591e-10
79.214	8.8614e-10
77.257	7.8902e-10
76.552	6.8137e-10
75.842	6.4385e-10
75.202	6.1808e-10
74.885	5.9815e-10
74.748	5.9531e-10
74.548	5.8961e-10
74.488	5.8547e-10

Details

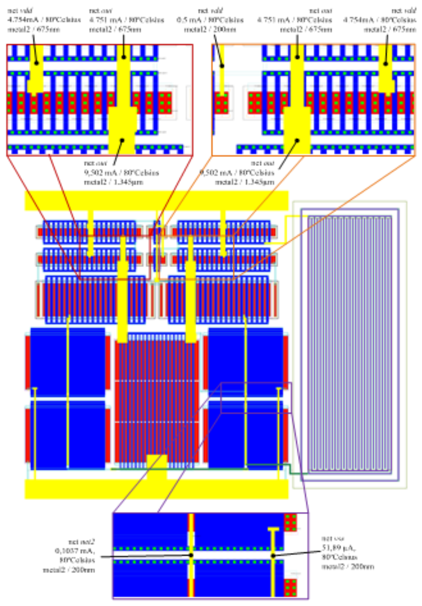
```
Summary:
    Failed constraints: 0

----------------------------------------
Design Variables
----------------------------------------
_l2g     = 8.0000e-07 close to lower limit [1.2E-7]
_lal     = 2.2000e-06
_lb      = 5.0000e-07 close to lower limit [1.2E-7]
_lc      = 1.3400e-05 close to lower limit [4.4E-6]
_lp      = 7.0000e-07 close to lower limit [1.2E-7]
_m2g     = 8.0000e+00 close to lower limit [1.0]
_mal     = 2.0000e+00 close to lower limit [1.0]
_mb2     = 1.0000e+01 close to lower limit [1.0]
_mbp     = 2.0000e+00 close to lower limit [1.0]
_mp      = 2.0000e+00 close to lower limit [1.0]
_nfc     = 4.4000e+01
_nrows2g         = 1.0000e+00 close to upper limit [1.0] LOWER LIMIT [1.0]
```

Router status : complete
25831/25856

Figure 12: AIDA’s Graphical User Interface navigating the sizing solution set and showing the corresponding floorplan and global routing.

The AIDA framework implements an automatic analog IC design flow from a circuit-level specification to a physical layout description. The circuit-level synthesis is done by AIDA-C, and after the circuit-level design, AIDA-L takes the device sizes and the best floorplan, and generates the complete layout, which is then saved as a GDSII stream format.

AIDA-C is a circuit-level synthesizer supported by state-of-the-art multi-objective optimization kernels, where the robustness of the solutions is attained by considering user-defined worst case corners, that account for process variations and(or) PVT corners. The circuit's performance is measured using Spectre®, Eldo® or HSPICE® electrical circuit simulators



. AIDA-L considers sized circuit to generate the complete layout by placing and, routing the devices, while fulfilling the technology design rules by using built-in DRC and LVS procedures. The router takes into account the circuit's currents to mitigate electro migration and IR-drop effects, and a fast but accurate PEX procedure provides parasitic estimates to be used in AIDA-C layout-aware optimization.

The framework's technology independent module generator, AIDA-AMG, is capable of creating several, simple and complex, device layout styles, allowing AIDA-C floorplan-aware sizing approach to explore a much wider space of solutions leading to higher quality layouts.

2. **ALADIN: A Layout Synthesis Tool for Analog Integrated Circuits**

BASIC ARCHITECTURE AND SYNTHESIS FLOW IN ALADIN:

The block diagram of ALADIN is depicted in Fig. 2. The entire system consists of three major components.

1) Module Generator Environment: It allows circuit designers to write technology- and

application-independent parameterizable modules as complex as needed

2) Design Assistant: It is integrated into Cadence Design Framework II (DFII) and can be easily ported to other commercial CAD software. It provides a graphical user interface (GUI) to help optimize an analog circuit from schematic to layout.

3) Technology Interface: It eases the input of description rules including the design rules, the electrical rules, and the special design constraints.

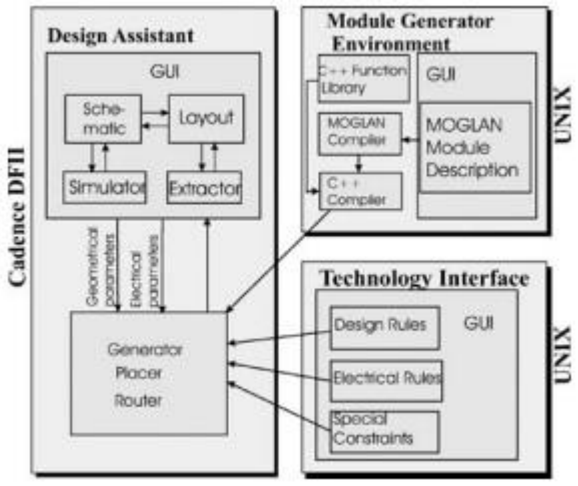


Figure 13: Block diagram in ALADIN

The circuit/layout synthesis using ALADIN is illustrated in Fig. 3. A simulated circuit is partitioned into several modules. The parasitic capacitance in the module layout is controlled by a capacitance sensitivity matrix, and the electrical rules, such as electro migration, matching, etc., are considered during the module generation/

The module selection and the layout generation are repeated until all the modules of the circuit are defined.

In the next step, the placement of the modules is performed simultaneously with the global routing. During the placement and the global routing, an appropriate topology of each module is chosen from all possible alternatives available in the design library. The layout synthesis flow is completed by the detailed routing with an embedded layout compaction capability. After the layout generation, an extraction is performed and the extracted parasitic elements are automatically annotated back into the schematic. The entire circuit is then simulated with accurate parasitic estimation, and the optimization loop starts.

<u>These feedback optimizations include three important Techniques:</u>

1) the application optimization that redefines the electrical constraints, such as the capacitance sensitivity matrix, to update the detailed module generation and the routing process;

2) the parameter optimization that changes circuit parameters and performs the layout generation for the impacted part of the circuit; and

3) the circuit architecture optimization that changes the structure of the circuit and, thus, undergoes a major update of the layout generation.

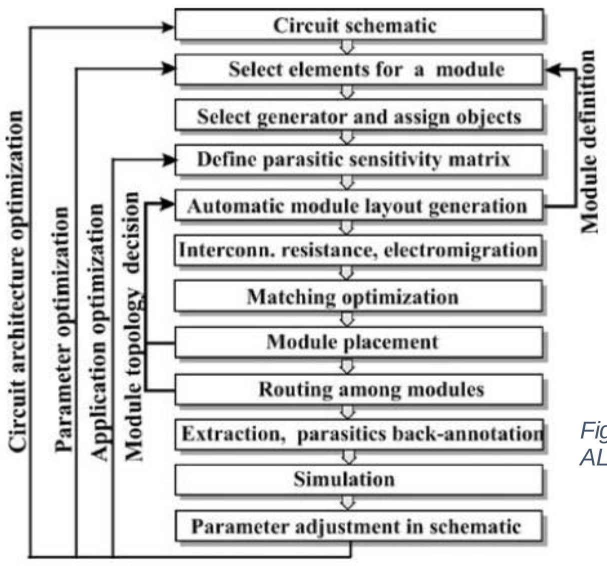

Circuit schematic
Select elements for a module
Select generator and assign objects
Define parasitic sensitivity matrix
Automatic module layout generation
Interconn. resistance, electromigration
Matching optimization
Module placement
Routing among modules
Extraction, parasitics back-annotation
Simulation
Parameter adjustment in schematic
Circuit architecture optimization
Parameter optimization
Application optimization
Module topology decision
Module definition

Figure 14: Circuit/layout synthesis using ALADIN

3. **LAYLA the Analog Layout Tool**

Fig. 4 gives an overview of a set of tools which have been developed to solve the analog circuit level layout generation problem. Together they provide an integrated framework, called LAYLA, for performance driven analog and mixed-signal layout design. Tools developed within the framework of this research are shown in straight lines, commercial tools that have been integrated in dashed lines.
We will pass upon all phases of the tool from circuit analysis reaching to routing and testability

The input of the tool set consists of three files:

1. Netlist file: The netlist describing the circuit for which the layout has to be designed this can be a device level netlist, or an architecture of higher levels blocks.

2. Specification file: A file with the specifications for the circuit.

3. Technology file: A file describing the process technology.

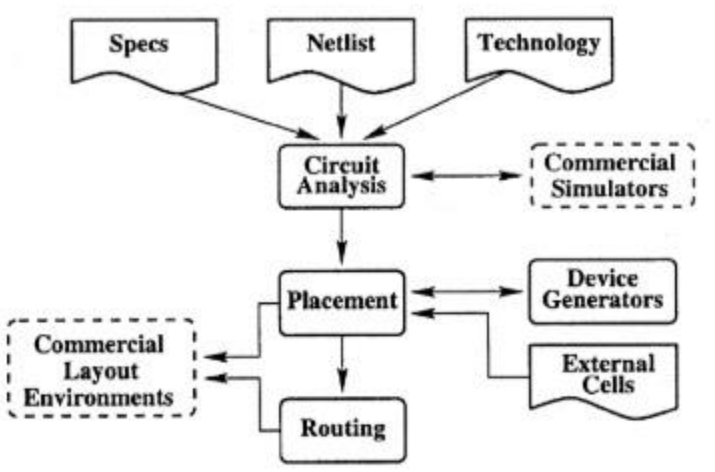


Figure 15: Software Architecture of LAYLA

Circuit Analysis

Before the actual layout process can be started, a number of numerical simulations has to be run to determine the necessary electrical information about the circuit. A circuit analysis program is used to automate this process.

The program interfaces is a commercial simulator. It generates a number of input files that runs the simulations and docs the post -processing of the simulator output. The most important task of the circuit analyzer is to extract the Sensitivities of the circuit performance characteristics with respect to all layout parasitics. The concept of sensitivities and how they are used to calculate performance degradation

Device Generation

A library of procedural device generators is used to create the layout tor basic devices (transistors, capacitors and resistors). The generators can be called in interface mode and in layout mode.
In interface mode they generate a list of possible layout variants for a device, based on its electrical parameters. Only the information necessary for the placement tool is generated:

the bounding box for the device and the terminal geometry.
After placement, the device generators are called again in layout mode to generate the complete layout for one selected variant. Special features have been added to the device generators, in order to make them suitable for use in a mixed-signal context.

<u>Placement</u>

The task of the placement tool in to select an optimal position, orientation and implementation for each device in the circuit, the freedom in placing these devices is used to control the layout-induced performance degradation within the margins imposed by the designer's specifications.
During each iteration of a simulated annealing optimization algorithm, the layout-induced performance degradation is calculated from the geometrical properties of the intermediate solution. The cost-function is designed to control performance degradation due to interconnect parasitics, mismatch and thermal effects.

<u>Routing</u>

The main task of the performance driven router is to route the circuit such that the performance degradation caused by the interconnect parasitic remains within the specification margins imposed by the designer
For a given set of circuit specifications, several valid routing solutions, can be found. Among these, the routing algorithm selects the solution that additionally maximizes the yield and the testability of the resulting layout. Initially, the circuit is routed with a cost function designed to enforce all performance constraints. After all nets have been routed, the layout parasitics are extracted and the performance of tile circuit is verified.
In a second phase, nets are pipped up and rerouted to optimize the yield and the testability of the layout

The following table 2.1 shows analog layout aware sizing tools

Table 2.4 Comparison between state-of-the-art works on layout-aware sizing with special emphasis to layout-related data included in-loop

	Layout Generator		Layout-related data included						
				Parasitics					
Work	Placer	Router	Geometric	Bulk	Intra-device	Inter-device	Interconnect	Resistances	Observations
Vancor. [99]	Procedural generator		✓	✗	✓	✗	✓	✓	1/2-D analytical-geometrical modeling of the critical nets; geometr data with equations
Pradhan [94]	Procedural generator (design space sampled only)		✓	✓	✓	✗	✗	✗	Analytical models for bu device and interconnect
Liao [96]	Template-based/ user-assisted	Channel router	✓	✗	✓	✓	✓	✓	Analytical models for ar and interconnect
Youssef [97]	Device-level procedural generator		✓	✓	✓	✗	✗	✗	Modeling of the stress effects of the devices
Ranjan [95]	Procedural generator (design space sampled only)		✓	✗	✓	✓	✓	✓	Area and interconnect using external extractor
Habal [103]	Enumeration of all possible floorplans	Exhaustive setup for Cadence Chip Assembly Router®	✓	✓	✓	✓	✓	✓	Complete extraction usir Cadence Assura®
Lopez [100, 101]	Coded slicing-tree	Template-based	✓	✗	✓	✓	✓	✗	3-D analytical-geometric
Berkol [102]	Layout description script		✓	✓	✓	✓	✓	✓	Complete extraction usir external extractor
This work, AIDA-L	Template-based Placer with multiple B*-trees	Automatic electromigration-aware WT and global routing in-loop	✓	✓	✓	✓	✓	✓	2.5-D modeling of the devices and routing that operates over non-detail routing

The following diagram shows a Chronological representation of analog layout generators.

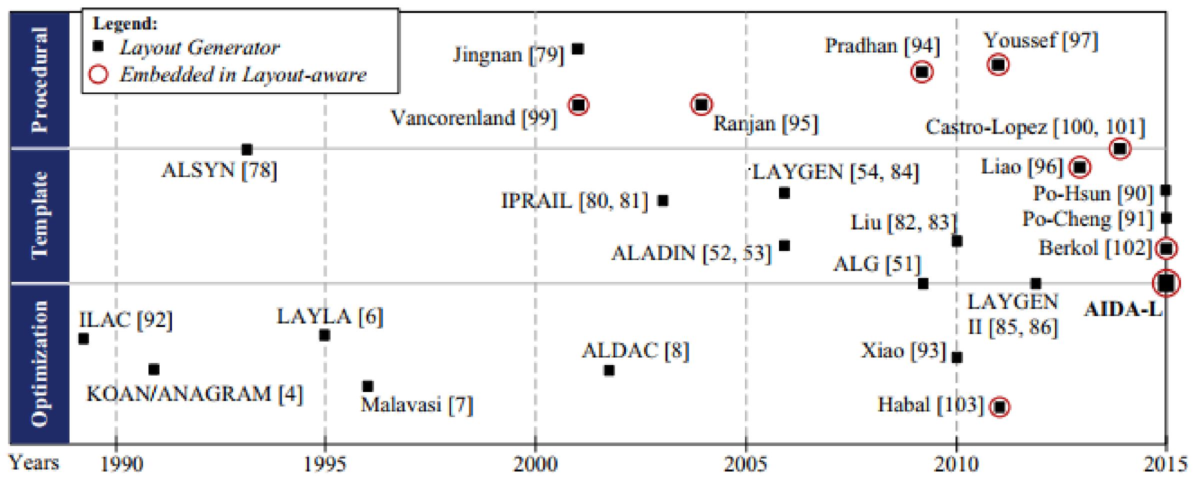


Figure 16: Chronological representation of analog layout generators

Conclusion:

A flexible module generator environment is developed, which allows designers to develop and maintain technology and application-independent module generators for relatively complex sub circuits.

A two-stage placement technique is proposed to dramatically improve the placement accuracy without compromising the computation efficiency compared to the one-stage placement. The analog module routing consists of the global routing and the detailed routing. The minimum-Steiner tree based global routing is integrated into the placement procedure to improve reliability and routability of the searched placement solutions.

The final layout is completed by a compaction-based constructive detailed router. The benefit of ALADIN is demonstrated by applying it to designing a number of circuits. Our technical analyses and experimental results have confirmed its competition over some existing automation tools and expert handcrafted layouts.

Chapter 3.
Tool Overview

3.1. Intro and features

ACML tool aims at automating the whole flow of creating a full current mirror layout (clean LVS and DRC Verification checks) guided by the user preferences to decide between wasted area and parasitics.
ACML is technology independent, it is optimized for small minimum feature length (λ) currently used like 7nm and 5nm. The layouts generated are of industrial quality, they are approved by experienced layout engineers inside the industry.

Features of Pattern Generator (Placement)

1- *Shared Source*
The user is given an option between sharing sources or not. This depends on the project requirements if the area is more important and we want to minimize parasitics then we choose to source share. However, if the area requirement is slightly relaxed and requirement on stresses being equal on all devices is more important, then the user may disable source sharing.

2- *Width Division*
Odd number of units represents a challenge for the quality of matching, 2 solutions are offered by ACML. First solution is to divide the width of the unit and thus doubling the number of all units and getting rid of odd numbers. This solution optimizes the use of area as it avoids the use of dummies. However it has other drawbacks, if units were already of small width, minimum width of the technology may be hit, also small units match worse because random mismatches are more significant (refer to matching section).Second solution is to get rid of those odd number of units by adding dummies. However this solution has another drawback as there will be a wasted area.

3- *Choosing the convenient pair*

ACML tool doesn't include floorplanning, however, it makes this process easier for the user. It provides a list of all possible pairs with their actual lengths and widths. These pairs are arranged according to their aspect ratio and this list states if routing will require any excess space for each pair.

4- *Good aspect ratio*

Sometimes if the total is a prime number, the only possible pairs have very bad aspect ratios, ACML solves this problem by adding one dummy device to get a total that has many factors, and thus many possible pairs with decent aspect ratios.

5- *Minimize the number of Devices on the same row*

Devices are arranged inside the pattern taking into consideration that we need to minimize the number of different devices in the same row for better routability.
Example:

Table 1: BAD for routability matching pattern example

F	B	D	E	C	G	No: 5
F	F	C	B	D	G	No: 5
B	E	A	A	C	G	No: 5
G	C	A	A	E	B	No: 5
G	D	B	C	F	F	No: 5
G	C	E	D	B	F	No: 5

Table 2: good for routability matching pattern example

B	**F**	**B**	**F**	**F**	**B**	No: 2
G	**C**	**G**	**C**	**C**	**G**	No: 2
D	**E**	A	A	**E**	**D**	No: 3
D	**E**	A	A	**E**	**D**	No: 3
G	**C**	**C**	**G**	**C**	**G**	No: 2
B	**F**	**F**	**B**	**F**	**B**	No: 2

6- *Similar number of Devices between all rows*

Sometimes number of different devices in a certain row is very large compared to the rest of rows, this causes waste of routes as the number of

routes placed on each row is determined by the largest number (maximum in all rows).
ACML tries to avoid this problem by making the number of different devices in all rows similar to each other.

Example:

Table 3: Matching pattern example 3

E	E	E	E	E	F	No:2
D	C	B	B	C	D	No:3
A	A	A	A	A	A	No:1
A	A	A	A	A	A	No:1
D	C	B	B	C	D	No:3
F	E	E	E	E	E	No:2

There is a waste in the routes in row 3 and row 4

Table 4: Matching pattern example 4

E	C	E	E	C	F	No:3
D	E	A	A	E	D	No:3
B	A	A	A	A	B	No:2
B	A	A	A	A	B	No:2
D	E	A	A	E	D	No:3
F	C	E	E	C	E	No:3

7- Putting the diode connected device as the reference

ACML's matching pattern puts into consideration that the diode connected device must be placed in the center of the pattern, as it is the reference device. The centroid of all devices must coincide with the centroid of this device.

Features of Router:

1- Area of routes is within the area of the mirror

We route all the route we need over the transistors and bulk connection se we don't take any extra area and this minimize the area needed for a current mirror and this decrease its area in the whole ship which give the designers a larger space to put other circuits

2- Routes are matched

We route equal number of routes over the transistors so all of them see the same resistance and capacitance so we don't affect the matching of the transistors and the extra rails are all connected together and also connected to the VSS in case Nmos and VDD in case of PMOS

3- Metal routes are convenient to the current flowing in them

To prevent electro migration we calculate each route width after knowing the current passes through it in order to be reliable and no current problems occurs.

4- Minimal Gate resistance

The tool we are working in it (synopsis tool) can place the transistor with all the gates connected together by polysilicon but from the training we took at silicon vision under the supervision of Eng. Fady we know that it is not good to connect between gates by poly as this puts a lot of resistance and capacitance on the gate as polysilicon has high resistance and capacitance per unit area

So to avoid this disadvantage and to make our product aligned with the industry we place the transistor without the default gate connection and we will make this connection to each finger alone then connect all the gates together with metal 1 so now the poly is not used to connect all gates

5- Clean LVS and DRC Verification

Our mirror passes the layout versus schematic test successfully and also basses all DRC rules.

3.2. USER GUIDE

i- <u>Step One</u>

Selection of required current mirror

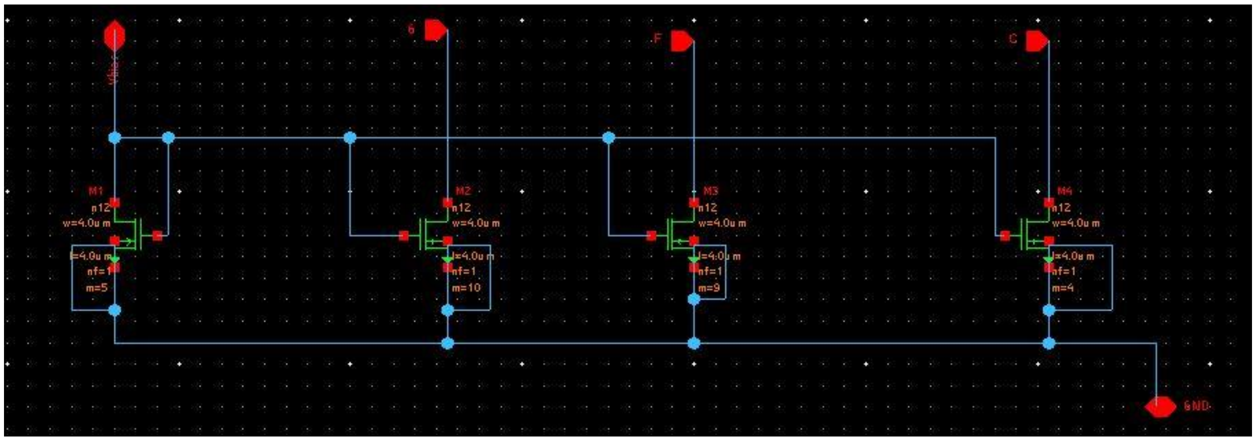
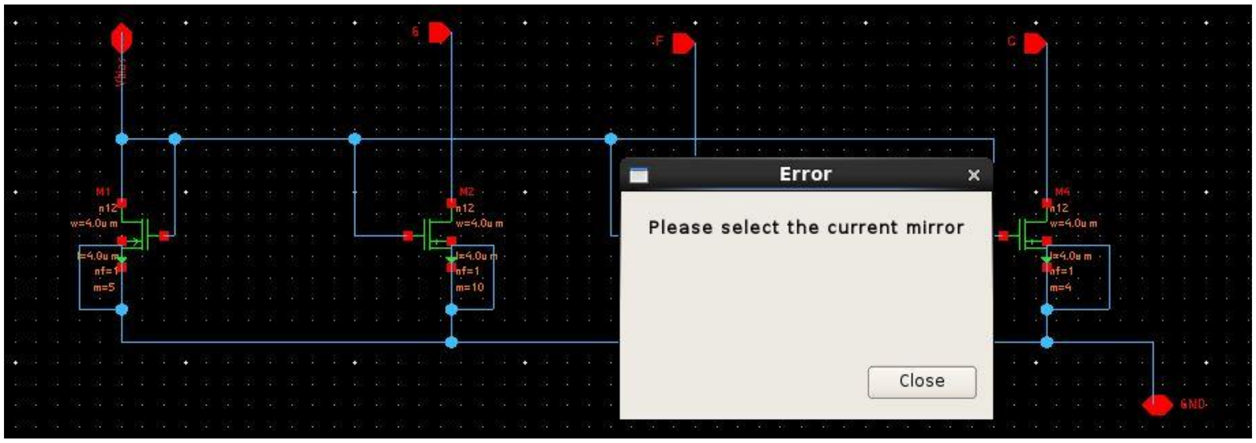


Figure 17: CM schematic Before Selection



Please select the current mirror

Figure 18: Tool's GUI interaction with user

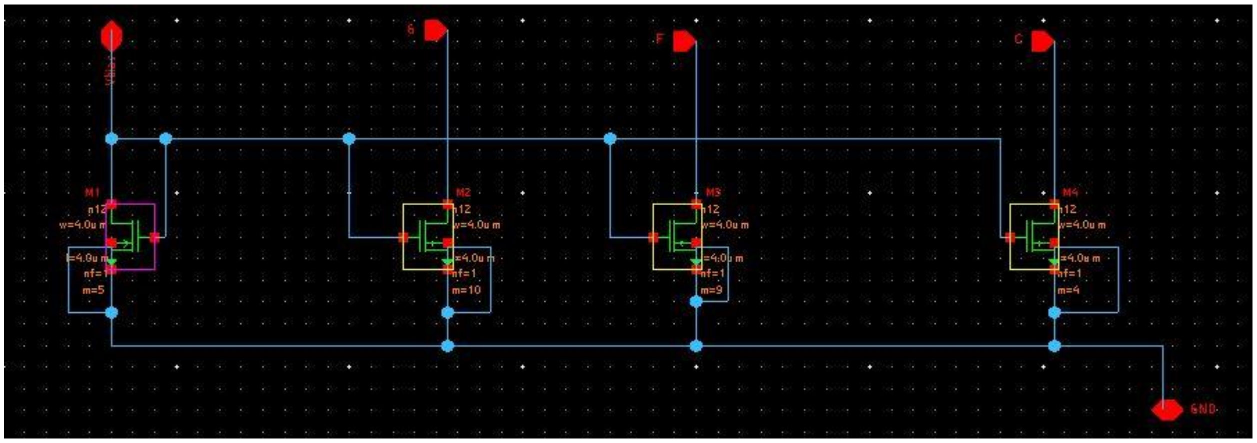
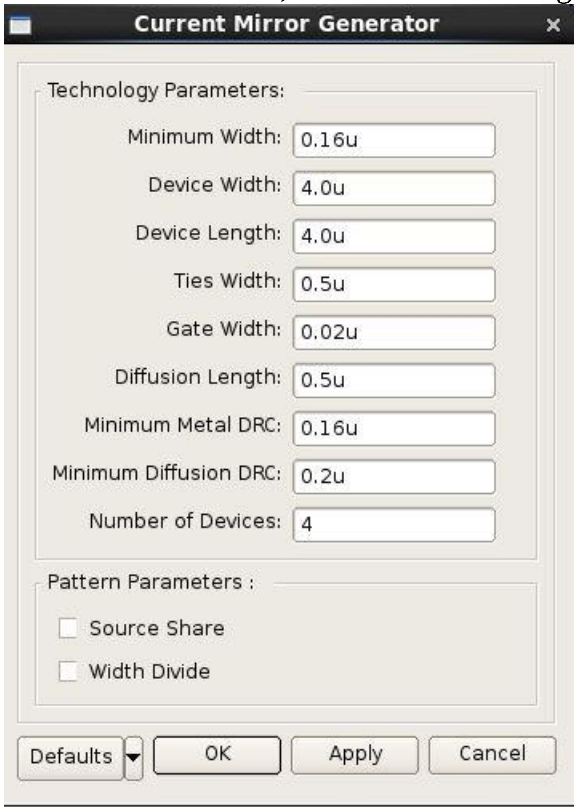


Figure 19: Tool's GUI response for user selection

ii- **<u>Step Two</u>**

After selection, a dialog box appears,



Current Mirror Generator

Technology Parameters:

Minimum Width:	0.16u
Device Width:	4.0u
Device Length:	4.0u
Ties Width:	0.5u
Gate Width:	0.02u
Diffusion Length:	0.5u
Minimum Metal DRC:	0.16u
Minimum Diffusion DRC:	0.2u
Number of Devices:	4

Pattern Parameters :

☐ Source Share

☐ Width Divide

Figure 20: Inputs for GUI

This dialog box lets the user provide the inputs required for the pattern generator

First category of inputs, are the technology dependents inputs which include:

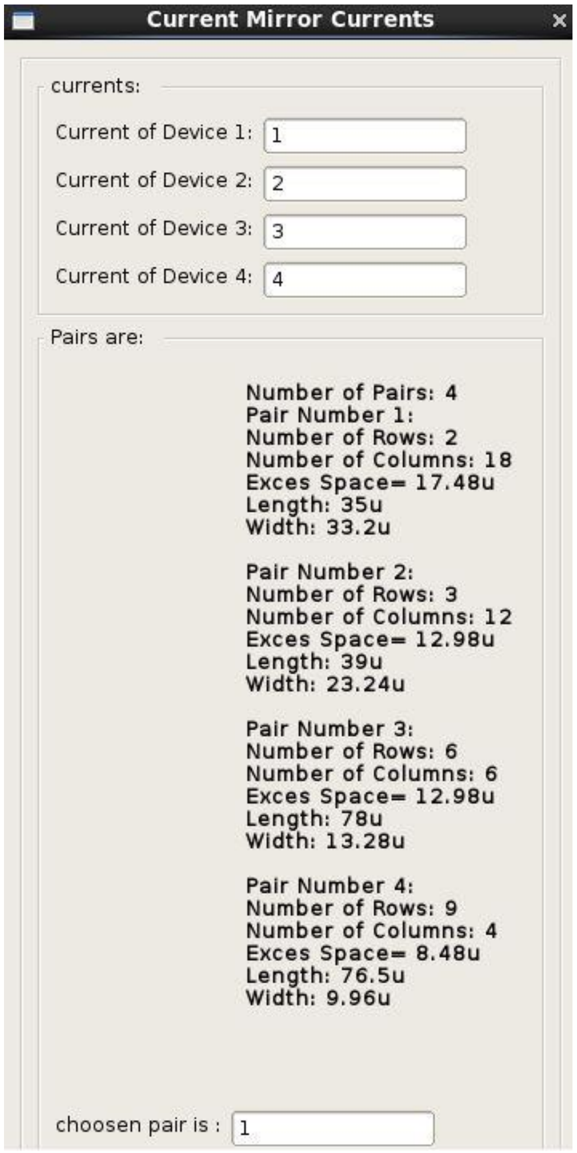
-Minimum width

- Ties width

-Gate width
-Diffusion length
-Minimum DRC between metal rails
-Minimum DRC between diffusion areas
Another category of inputs are inputs that depend on the current mirror selected which are:
-Finger width
-Finger length
-Number of devices
And the last input is passed automatically to the pattern generator which is the number of units for each device (number of fingers * number of multipliers)

Last category of inputs is the features inputs which are:
-Source sharing is applied or not
-Width division is allowed or not (Dummies are allowed or not): not allowing width division implies the use of dummies

iii- Step Three



currents:

Current of Device 1: 1

Current of Device 2: 2

Current of Device 3: 3

Current of Device 4: 4

Pairs are:

```
Number of Pairs: 4
Pair Number 1:
Number of Rows: 2
Number of Columns: 18
Exces Space= 17.48u
Length: 35u
Width: 33.2u

Pair Number 2:
Number of Rows: 3
Number of Columns: 12
Exces Space= 12.98u
Length: 39u
Width: 23.24u

Pair Number 3:
Number of Rows: 6
Number of Columns: 6
Exces Space= 12.98u
Length: 78u
Width: 13.28u

Pair Number 4:
Number of Rows: 9
Number of Columns: 4
Exces Space= 8.48u
Length: 76.5u
Width: 9.96u
```

choosen pair is : 1

Figure 21: GUI first output to user

This dialog box requires the user to provide the currents that pass in each device, this piece of information is passed to the router to let it avoid electron migration.

Also, in this dialog box, Pattern generator displays all the possible pairs (combinations of rows and columns) and lets the user choose the convenient pair according to the floorplan decided by the user.

To ease this process for the user, actual length and width of each pair are provided, in addition to the excess space added for routablilty if any is needed.

iv- <u>Output:</u>

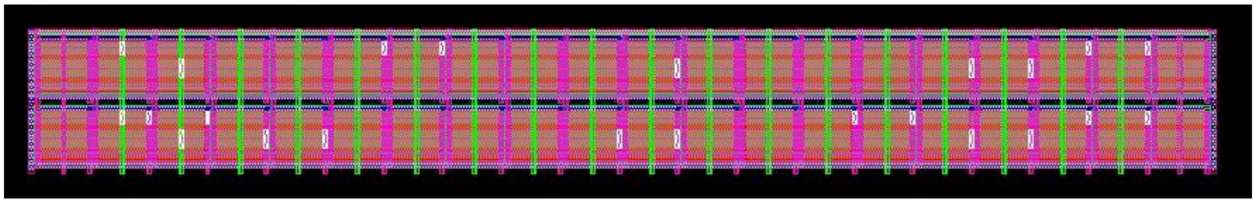


Figure 22: Final tool's output, fully routed CM

Output includes a fully placed and routed matched current mirror which are ready for passing the verification step (clean LVS and DRC checks).

Chapter 4.
Common centroid matching

4.1. Matching in Analog layout

Matching is one of the main challenges in Analog layout, that doesn’t arise in digital circuits. The presence of differential signals and reference currents makes analog circuits very sensitive to mismatches. Mismatch and parasitics induced by layout can greatly degrade the performance of analog circuits.

Mismatch is defined as the process that causes process induced, time-independent random variations in physical qualities of identically designed devices . Nowadays, since the functionality of most analog circuits is based on relative characteristic parameters of analog devices rather than absolute value of those, mismatch puts a fundamental limit on the achievable circuit performance in a particular technology process.

Matching of device characteristics such as the threshold voltage Vt is important. Circuits using these structures with device threshold differences of a few millivolts or less can determine the performance and yield of a design.

The threshold difference between a pair of (otherwise identical) MOS devices is due to the variations in number of doping atoms in the channel. This difference has been shown to be proportional to the inverse square root of the channel area, and it reduces with decreasing gate oxide thickness.

Poorly matched devices (transistors, capacitors, resistors) can lead to non-idealities

-Amplifier Offset

-Converter Nonlinearity

-Gain Error

1. Sources of matching errors

<u>1-Systematic Mismatch</u>

Introduced by circuit/layout designer and can usually be avoided
-Some good design techniques exist to help minimize these matching errors:
-use multiples of small, unit sized devices (transistor stripes, resistors and capacitor areas)
-use cascodes- increased output impedance (smaller current variations with changes in VDS)
-Avoid asymmetric loading- especially for dynamic signals (match wire lengths, capacitances)
-Don’t mix different type of devices if they’re not supposed to match (e.g poly resistors and n+ resistors)

<u>2-Random Mismatch</u>

Variation in Process Parameters and lithography and it’d beyond the designer’s control but he must take these into account during the design process
They are due to random variation in
-Device length
-Channel Doping
-Oxide thickness
-Sheet resistance
-Capacitance
These errors are reduced by:
-Increased device area
-Increased area/perimeter ratio (square is best)
In a rectangular device with active dimensions W by L, an aerial mismatch can be modeled as:

$$\sigma(P) = \frac{k_P}{\sqrt{WL}}$$

3-Gradient Mismatch
First or second order fluctuations over longer lengths across the chip
-To avoid these errors, devices should have similar environment
Same size, orientation, location, supplies, temperature
-Minimize these errors with some layout techniques
Common centroid- when devices are supposed to be matched, balance them so that their centroids are the same (eliminates 1st order gradient errors)
Interdigitization- not strictly common centroid but reduces impact of gradient errors.
In theory two device with the same size have the same electrical properties.

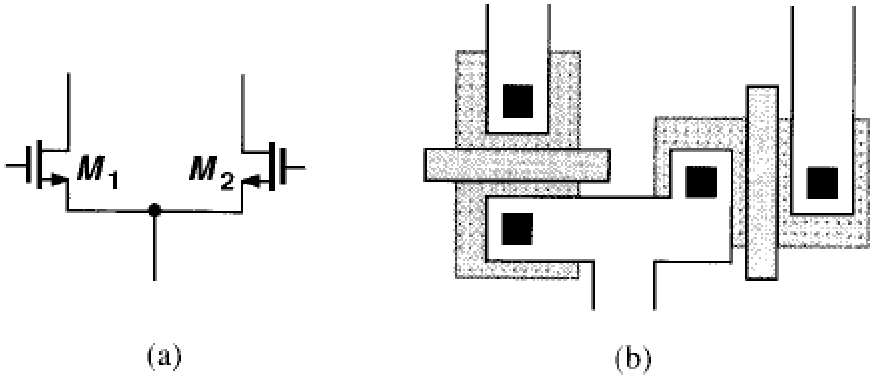


Figure 23: Gradient Mismatch

In reality there is always **process variations**:
1-Silicon is anisotropic:
•Ion implantation is performed at an angle causing shadow
•Source and drain may not be symmetric due to ion implantation angle,
–necessary to avoid implant depth issues (channeling).

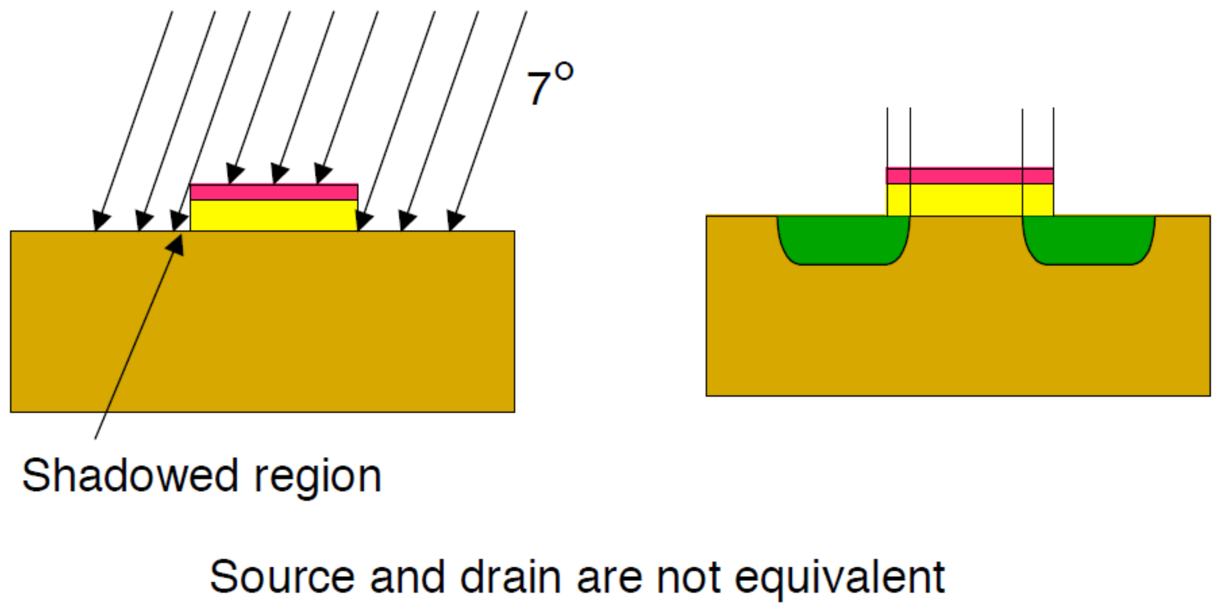


Figure 24: Process variations

<u>2-Photo-lithographic invariance (PLI):</u>

•Lithography effects are different in different direction.
•Orientation is important in analog circuits for matching purposes
–C and D are better
–Maintain orientation
C. Gate aligned

D. Parallel gate:

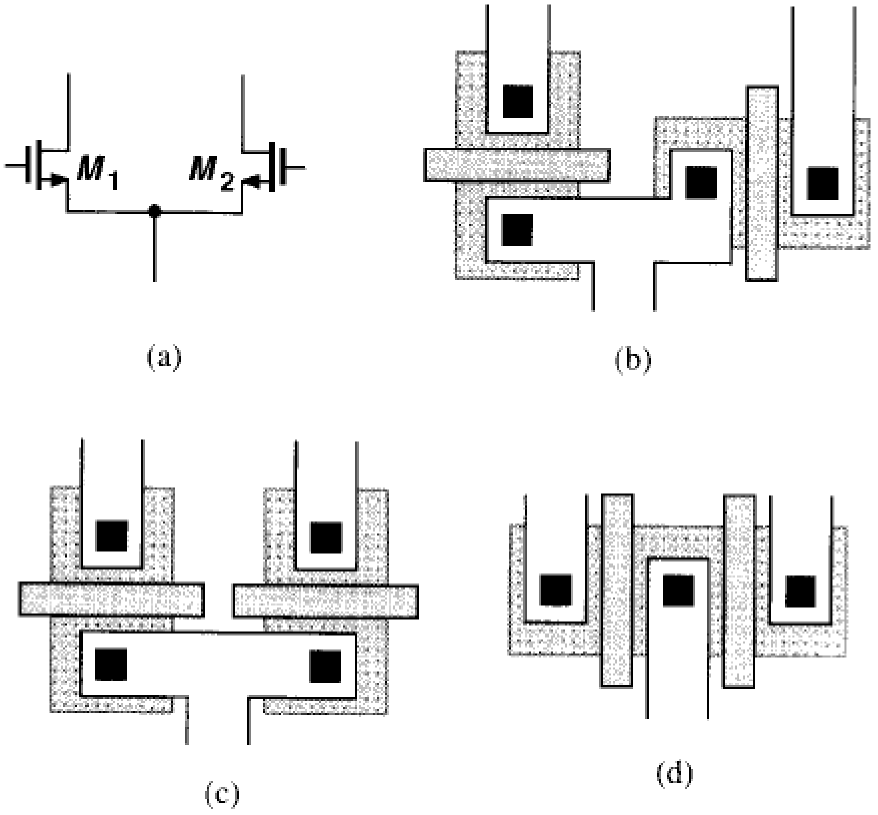
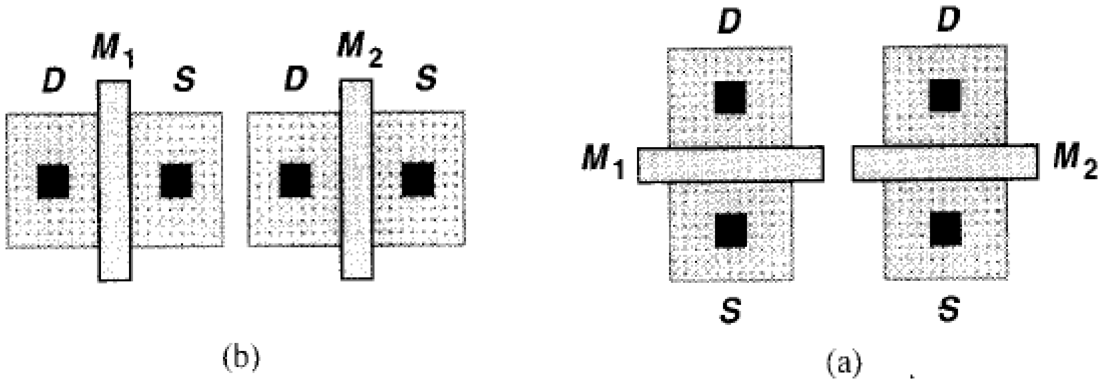


Figure 25: Matching

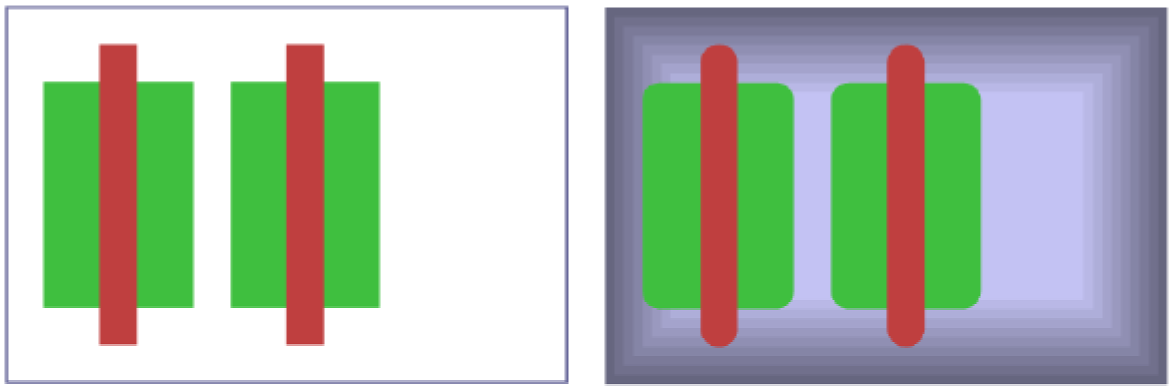
–Two drains have different surroundings
–Two sources have different surroundings
•Current flows in the same direction



3-Well Proximity Effect

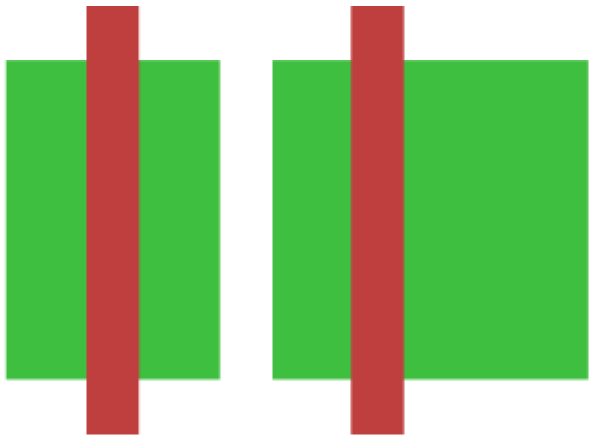
High energy ion implants to form the well.
• Scattering from the edge of the photoresist mask, and embedding in the silicon surface (near well edge).

• Transistors close to the well edge will therefore have different properties.
• This is known as the well proximity effect (WPE). Important for matching.



As with S/D,
implantation angle may
render the scattering
and doping asymmetric

Shallow trench isolation strains the active area of the transistor. Influences mobility and threshold voltage (stress induced enhancement or suppression of dopant diffusion).
•Distance between gate and STI impacts performance.
•Important for matching.



These effects include Vt shifts resulting from the proximity of the gate to the N-well edge and Vt 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.

Matching of Devices allows circuit design based on ratios of values.

2. Rules of optimum matching

Devices to be matched should have:
1. Same structure
2. Same temperature
Negligible dissipation on chip:
• no problem for constant ambient temperature T
• possible small DT at fast transients of ambient T.
• Heat generation on chip:
• place devices to be matched
- close to each other
- as far as possible from the source of heat
- on the same isotherm
- exploit the symmetry axis of the chip

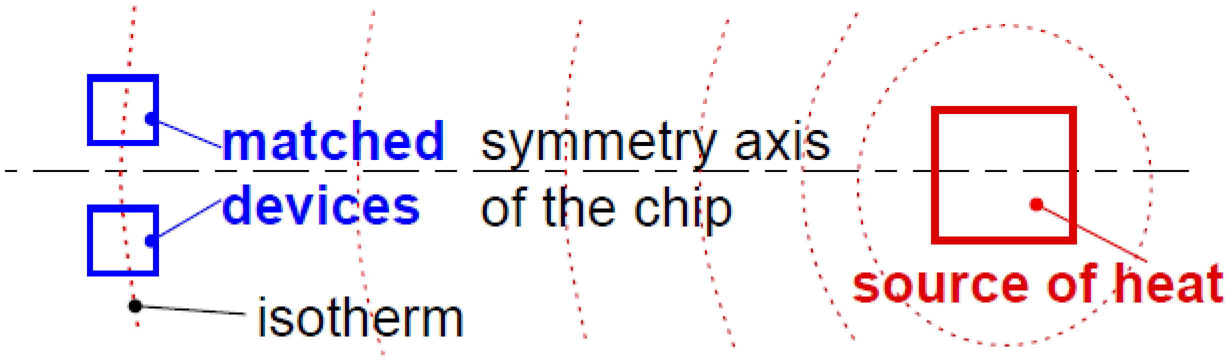
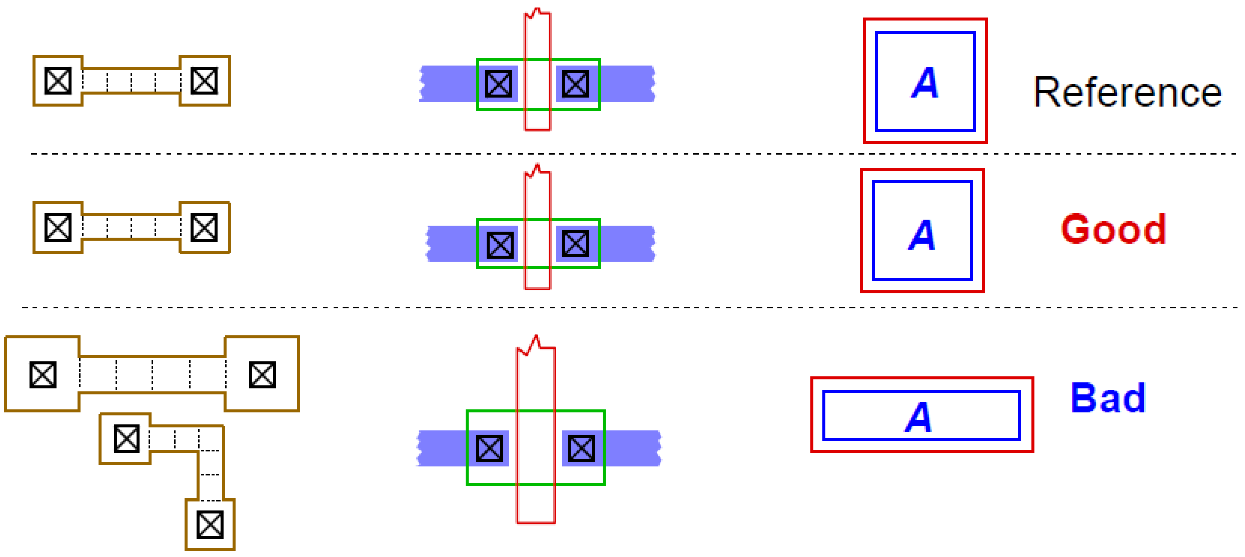


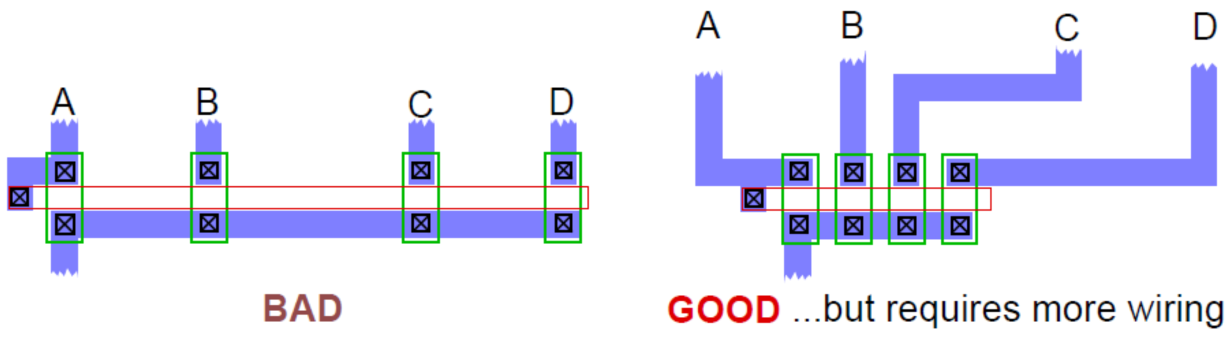
Figure 26: Temperature gradient

3. Same shape and same size

-Resistors	Transistors	Capacitors
(4 squares)	(W/L=2)	(area A)



4. Minimum distance
To take advantage of spatial correlation of fluctuating parameters
Example: Multiple Current Mirror



5. Common centroid geometries
6. Same orientation on the chip
As shown in Figure 1, a MOS
transistor has an asymmetrical structure due to its fabrication, and it causes current direction dependent mismatch. Hence, a symmetrical placement that matches current direction (as
in the layout in Figure 1(b)) is preferable.

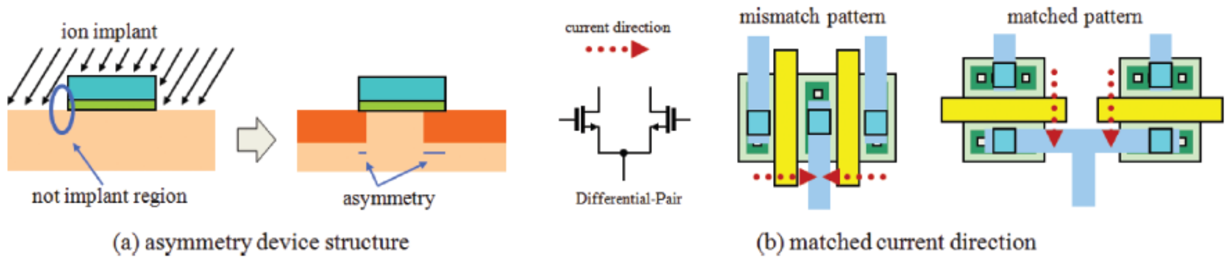
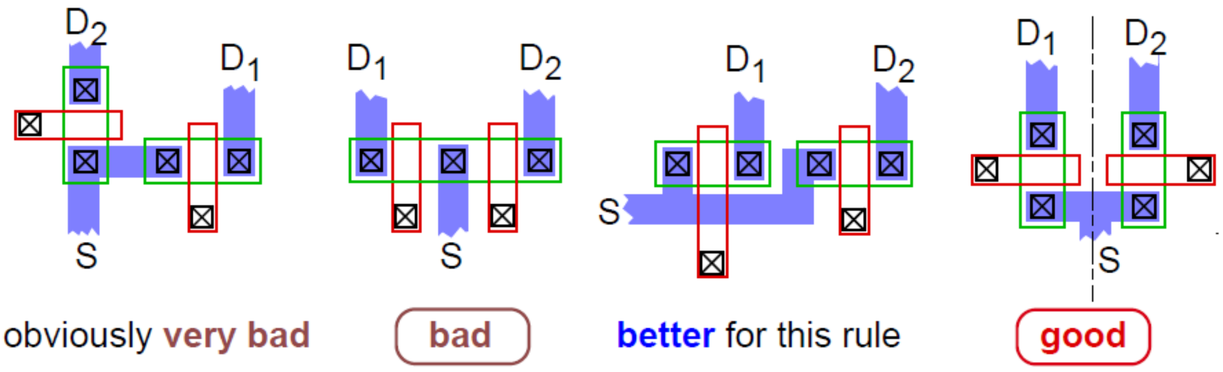


Figure 27: Asymmetry device structures and matching current directions

To eliminate possible differences due to:
• Anisotropic substrate
• Anisotropic process steps
• Stress before and/or after packaging.
• Example: implementation of a differential pair.



7. Same surroundings (same neighbourhood)
Various possible reasons, not always clear

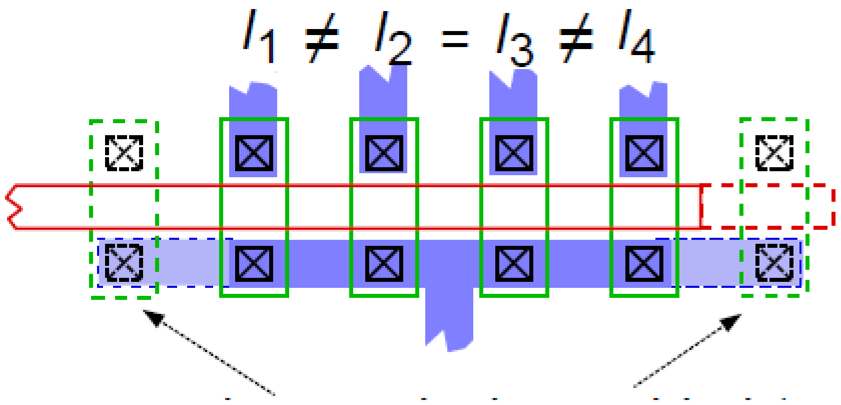
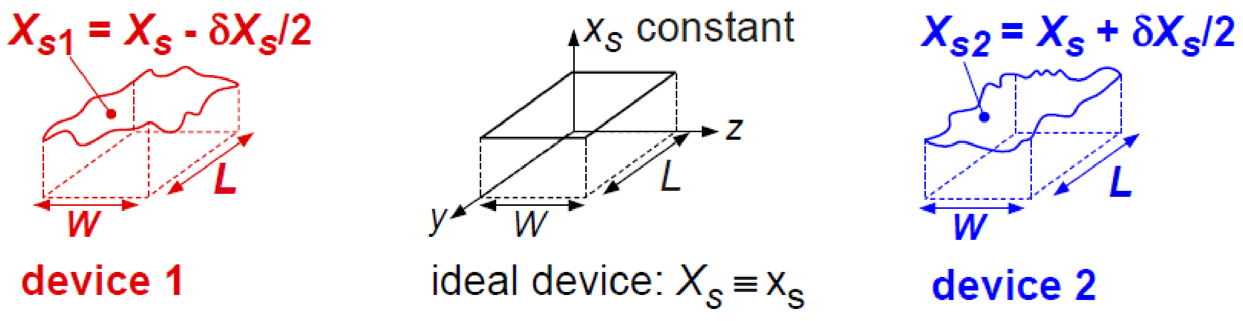


Figure 28: Dummifiaction

dummy devices added (not used in circuit)
• This "end effect" is also found in 2-D arrays
(first and last rows and columns different from rest of array)
LA-17
• May be ensured by implementing dummy structures.
• Example: multiple current source:
• If the reason is known, surroundings can be simulated by an adequate structure
8. Non-minimum size.

a. Effect of uncorrelated spatial fluctuations of specific parameter *xs*

(for example *tox*)
Definition: *Xs* = spatial average of *xs*(*y*,*z*)

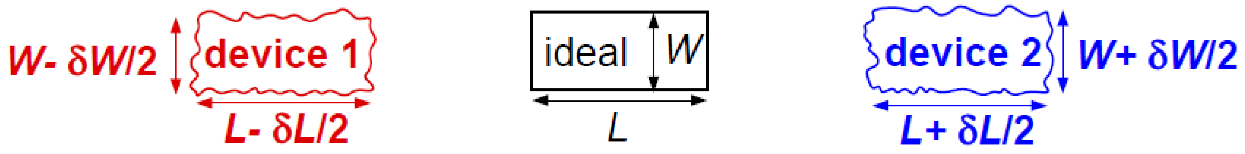

$X_{s1} = X_s - \delta X_s/2$
W
L
device 1
x_s constant
z
y
W
L
ideal device: $X_s \equiv x_s$
$X_{s2} = X_s + \delta X_s/2$
W
L
device 2

It can be shown that:

$$\boxed{\sigma(\delta X_s) \propto 1/\sqrt{WL}}$$

(statistical averaging of xs over area WL)

b. Effect of uncorrelated periphery fluctuations:



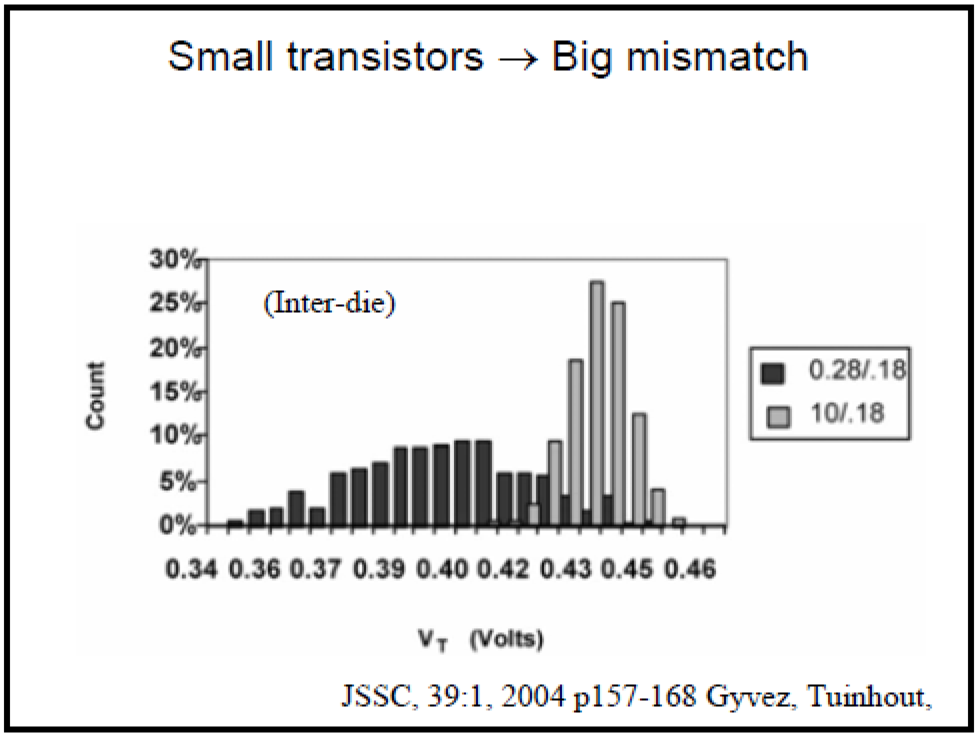
Now: s(d*W*)µ1/ *L* width *W* is averaged along length *L*

s(d*L*)µ1/ *W* length *L* is averaged along width *W*

and relative mismatch: d(*WL*±1)/(*WL*±1) = d*W*/*W* ± d*L*/*L* +: area *WL*

–: ratio *W*/*L*

$$\boxed{\sigma\left[\frac{\delta(WL^{\pm1})}{WL^{\pm1}}\right] \propto \sqrt{\frac{1}{W^2L}+\frac{1}{L^2W}} = \frac{1}{\sqrt{WL}}\sqrt{\frac{1}{L}+\frac{1}{W}}}$$



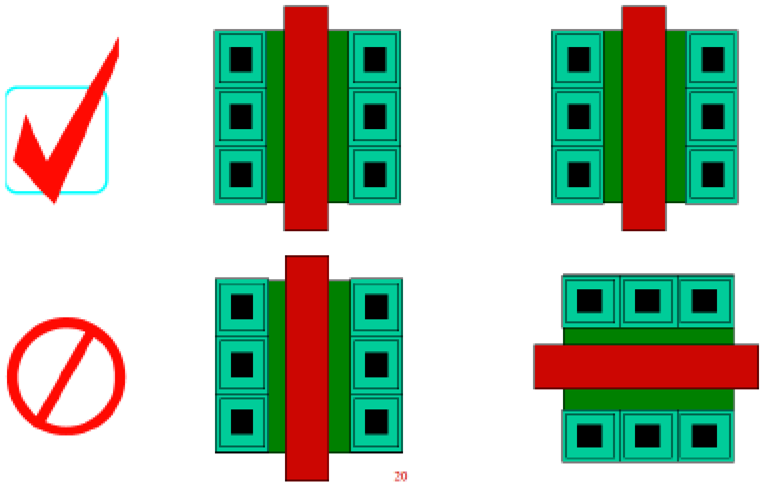
Small transistors → Big mismatch

JSSC, 39:1, 2004 p157-168 Gyvez, Tuinhout,

Figure 29: Transistor size vs Mismatch

3. Geometric effects

* Large transistors match better than small transistors
– Fluctuations average out over larger area
* Long channel transistors match better than short
– Less Channel length modulation effects
* Identically oriented transistors match better
– Silicon is anisotropic and hence has different conductance in different directions
*Transistors with thinner gate oxides match better (assuming same area)
– Higher back gate doping fluctuations average out over larger area
– Edge fringing effects are less pronounced
* Transistor orientation is important



Etch Effects

* Polysilicon does not always etch uniformly
– Large openings etch faster than small openings in mask
– Solution is to use dummy structures

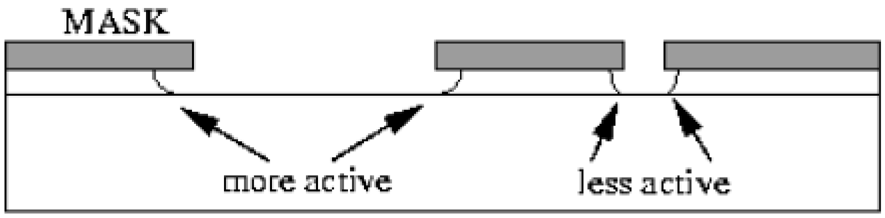
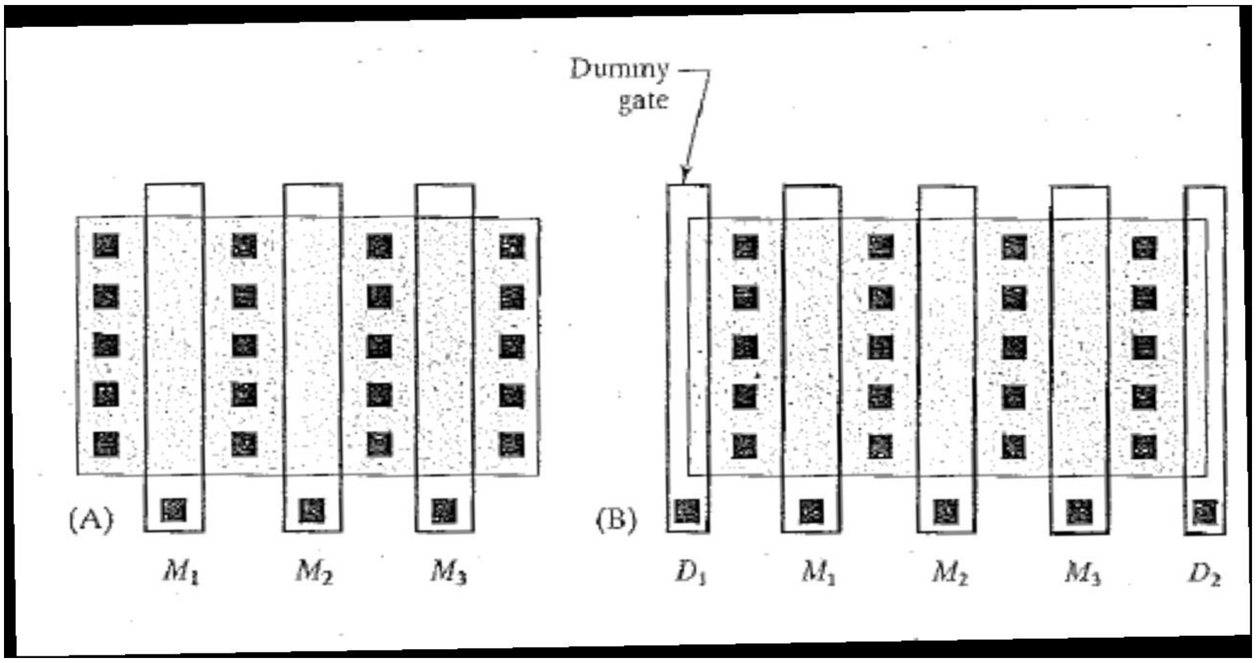


Figure 30: Etching process



4. Diffusion effects

* Diffusion widens implanted region
– Can affect doping of neighboring devices
– Solution is to increase distance and use dummy structures that affect all transistors the same

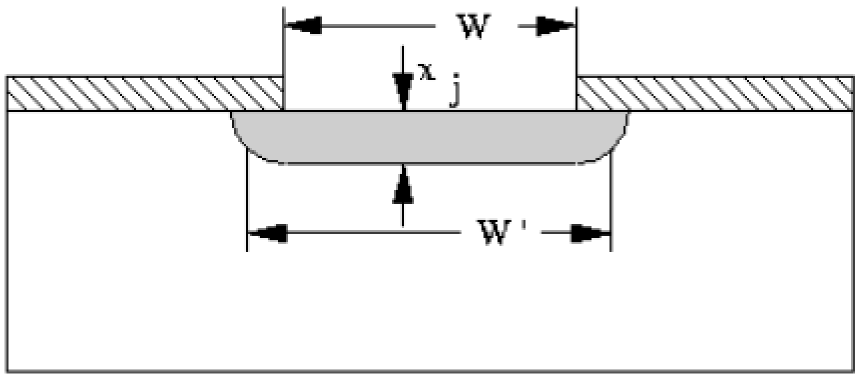


Figure 31: Diffusion effect

5. Thermal effects

* Temperature affects
– Mobility and threshold voltage
– Resistance value

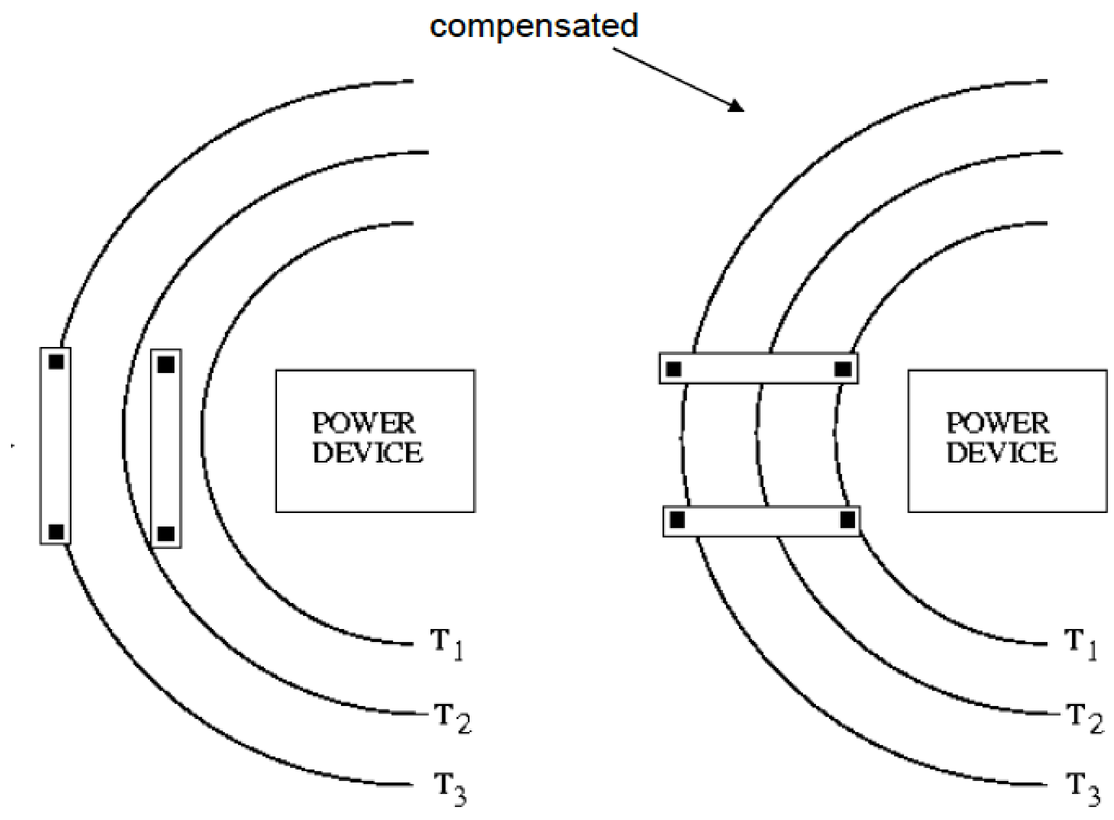
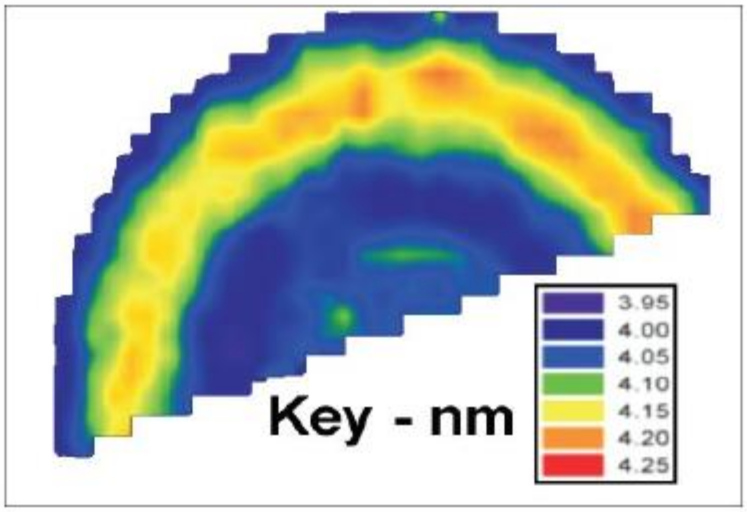


Figure 32: overcoming temperature gradient

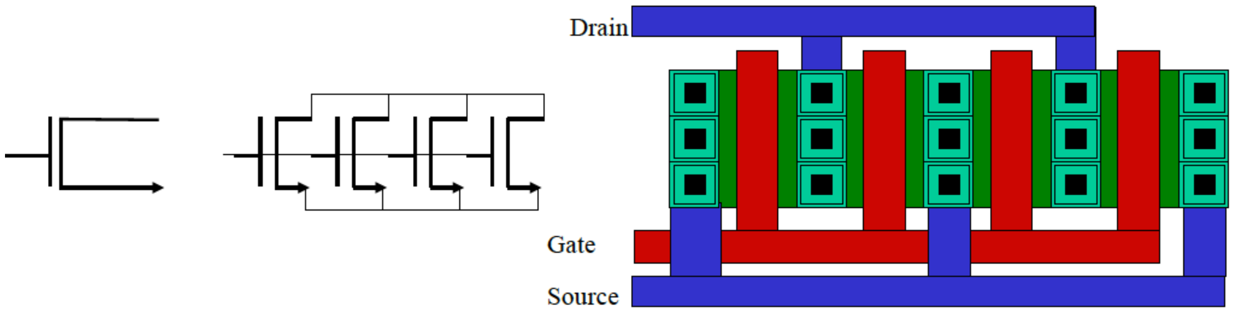
6. Stress effects

* The fabrication under high temperatures may leave residual stresses in chip
* Packaging can cause stress in chip Solutions
* Keep critical matched devices in center of chip or on centerlines
* Avoid using corners for matched devices Oxide Thickness Gradients
* Thermally grown oxides depend on temperature and oxidizing atmosphere
* Modern oxidation furnaces, although well controlled in temperature still have temperature gradients.



Oxide thickness on 200 mm wafer

Dealing with Large transistors
These can be split into many parallel fingers
* Contact space is shared amongst transistors
* Parasitic capacitance is reduced



4.2. Common centroid matching

For devices that do require matching, current mirrors, differential pairs, etc., each device is broken up into fingers of the same width, which are interdigitated together to form a compound device.

Usually, for two devices of the same size that share a source or drain, a common-centroid layout with dummy devices is used. If more devices must be matched, an interdigitation pattern with dummy devices is used.

In a simple current mirror usually requires two perfectly matched transistors. But there is always some mismatch between these two transistors after fabrication and the circuit performance will be degraded seriously. In general, the mismatches caused by manufacturing processes can be classified into systematic mismatch and random mismatch. The random mismatch can only be reduced by increasing layout area, but the systematic mismatch can always be alleviated through proper layout techniques.

Any parameter variation would alter the characteristics of transistors and their applications. Among lots of transistor parameters, the variation in threshold voltage has the most serious impact on transistor currents. Nowadays, integrated circuits are getting more complicated. The dual matching layout techniques no longer satisfy all of the high performance integrated circuits.

In the following, the main layout techniques to minimize unwanted effects caused by systematic, and gradient mismatch in analog ICs are listed [3]. These techniques apply to all elements in a group of n devices which have to be matched.

That is, they have to realize intended ratios of their electrical parameters with an optimal precision.

• Minimize distances between the elements (suppresses gradient mismatch)

• Split devices into same-sized sub-devices (suppresses fringe effects)

• Interdigitate sub-devices of different devices (improves suppression of gradient mismatch)

• Place elements in the same direction (synchronizes direction of current flow)

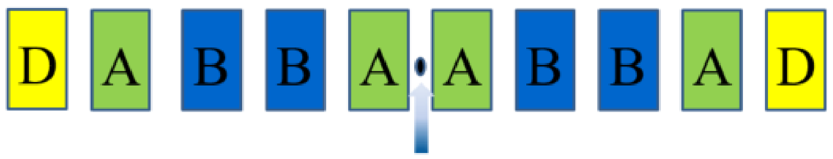
• Place elements symmetrical (additionally improves suppression of gradient mismatch)

• Add dummy elements around the group (suppresses fringe effects caused by different neighboring elements)

• Increase size of basic elements (minimizes fringe effects due to better area-boundary-ratio, but leads to increasing gradient effects)

Common matching techniques are interdigitated layout and common centroid layout. It can be shown that these techniques reduce the effect of parameter variations (gradients) across the die.

Common Centroid matching technique: placing components such that all components

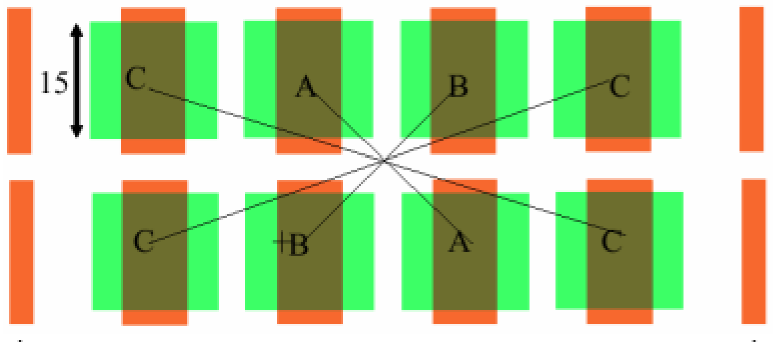


have the same centroid.

The common centroid technique is very good at reducing the effect of thermal or linear process gradients that may be present in an integrated circuit. This technique distributes the effect of gradient more evenly among the devices (assuming that within any small piece of silicon area the process and temperature gradient is linear). A thermal gradient, for instance, is generated by a hot spot on the chip that can change the electrical characteristics of a device. Devices close to the hot spot will be affected more than devices that are further away. This technique distributes the gradient effect more evenly among the devices.

Gradient-induced mismatches can be minimized by reducing the distance between the centroids of the matched devices. Some types of layout can actually reduce the distance between the centroids to zero. These common-centroid layouts can entirely cancel the effects of long-range variations as long as there are linear functions of distance. Even if the variations contain a nonlinear component, they still remain approximately linear over short distances.

The more compact the common centroid layout can be made, the less susceptible it becomes to nonlinear gradients.



Common centroid structure: is a sequence or a matrix that is composed of device fingers and satisfies the following conditions:

- Coincidence: The centroid of the different matched devices should at least approximately coincide. Ideally, the centroids should exactly coincide, which is called the completely common centroid structure.
- Symmetry: The structure should be symmetric around both X-axe and Y-axe.
- Dispersion: The fingers of each device should be distributed throughout the structure as uniformly as possible.
- Compactness: The structure should be as compact as possible. Ideally, it should be nearly a square.
- Orientation: Each matched device should consist of an equal number of segments oriented in either direction.

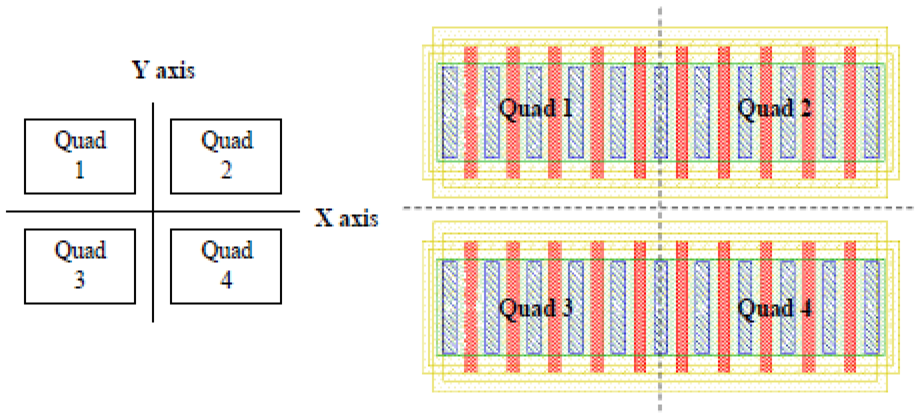


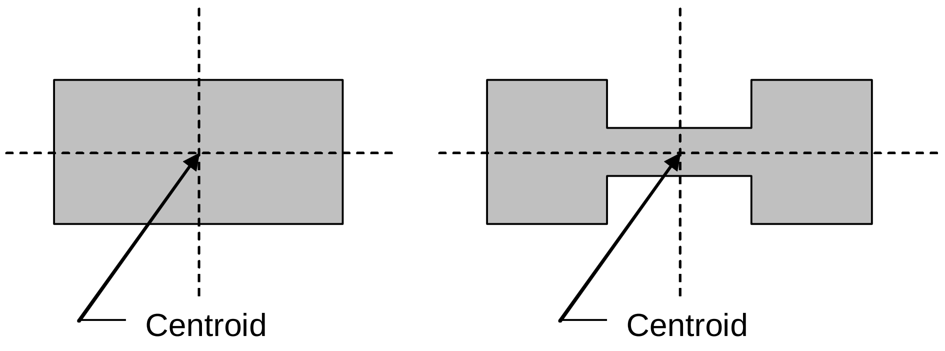
Figure 33: Cross-Quad matching

How to Find a Centroid (Easily)

Rules for finding a centroid (assuming linearity):

If a geometric figure has an axis of symmetry, then the centroid lies on it.

If a geometric figure has two or more axes of symmetry, then the centroid must lie at their intersection.



The centroid of an array can be computed from the centroids of its segments.

If all of the segments of the array are of equal size, then the location of the centroid of the array is the average of the centroids of the segments:

$$d_{array} = \frac{1}{N}\sum_{N} d_{segment}$$

Note that the centroid of an array does not have to fall within the active area of any of its segments.

This suggests that two properly constructed arrays could have the same centroid….

Arrays whose centroids coincide are called common-centroid arrays.

Theoretically, a common-centroid array should entirely cancel systematic mismatches due to gradients.

In practice, this doesn't happen because the assumptions of linearity are only approximately true.

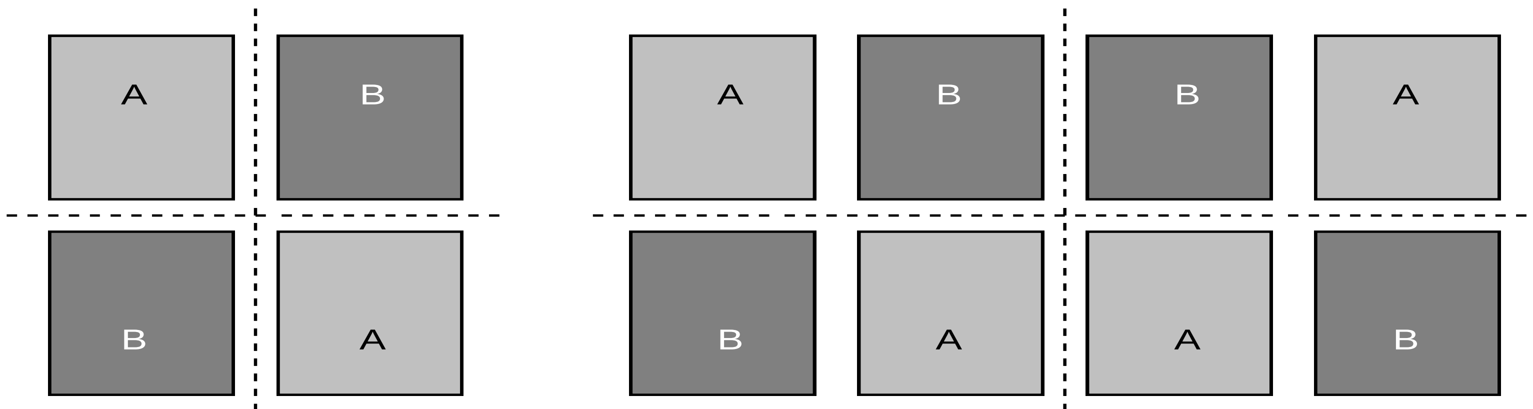
Common-centroids don't help random mismatches at all. Neither are they a cure for sloppy circuit design!

Virtually all precisely matched components in integrated circuits use common centroids.

2D Cross-coupled Arrays

A more elaborate sort of common-centroid array involves devices cross-coupled in a rectangular two-dimensional array.

This type of array is ideal for roughly square devices, such as capacitors and bipolars.

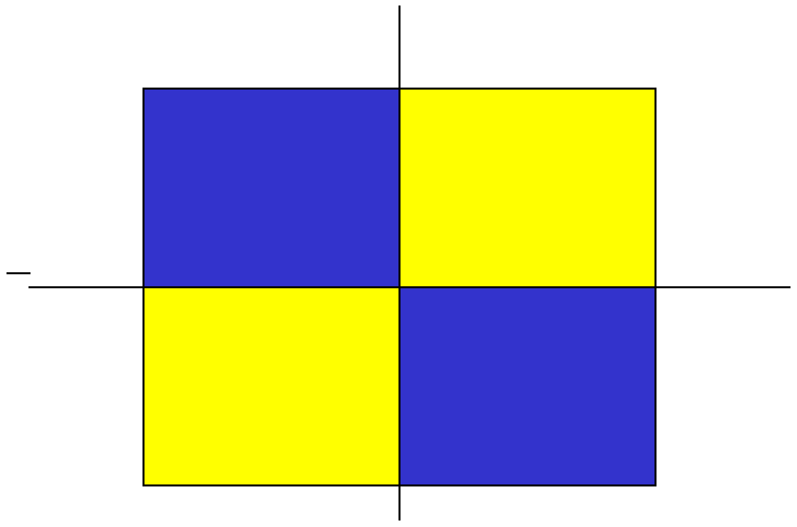


The simplest two-dimensional cross-coupled array contains four segments.

This type of array is called a cross-coupled pair.

For many devices, particularly smaller ones, the cross-coupled pair provides the best possible layout.

More complicated 2D arrays containing more segments provide better matching for large devices because they minimize the impact of nonlinearities.



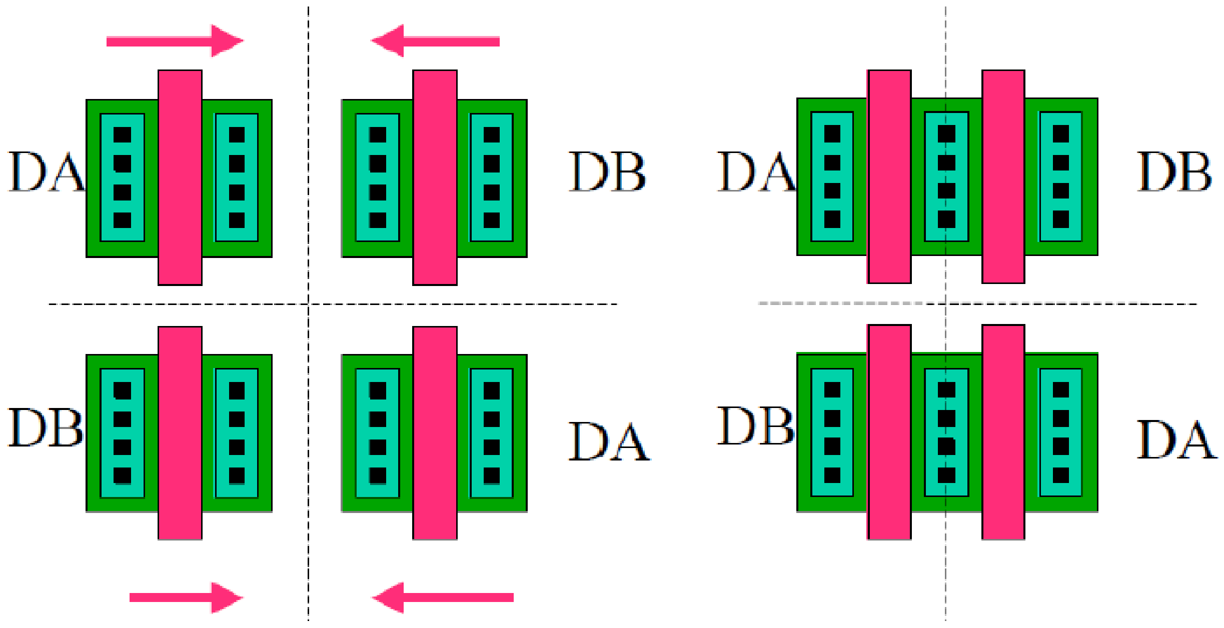
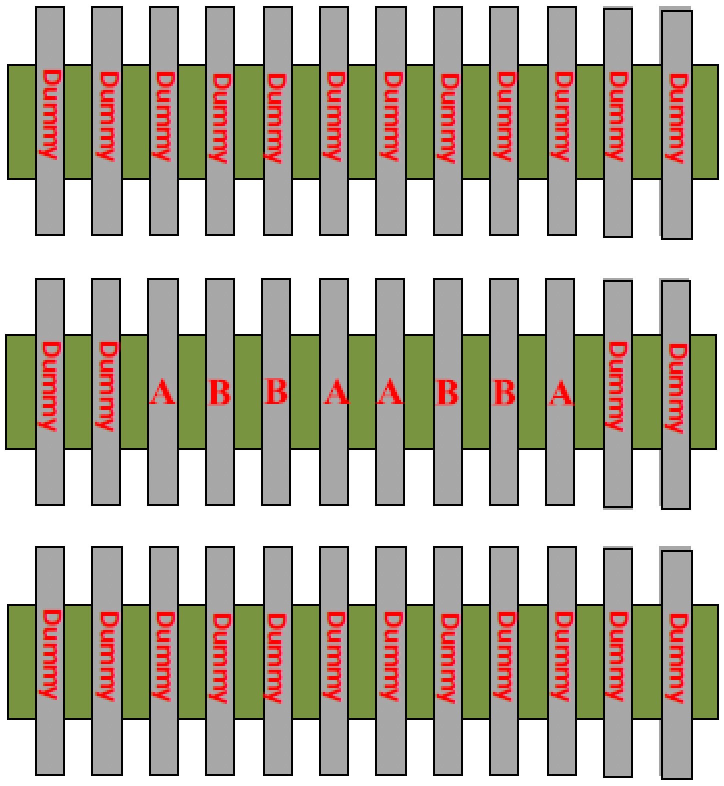


Figure 35: Current flow and matching

Use of Dummy Devices (Dummification)

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



the

Figure 36: common centroid CM array protected with dummies

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.

Substrate Connection

Another critical parameter for matching nanoscale devices is the substrate

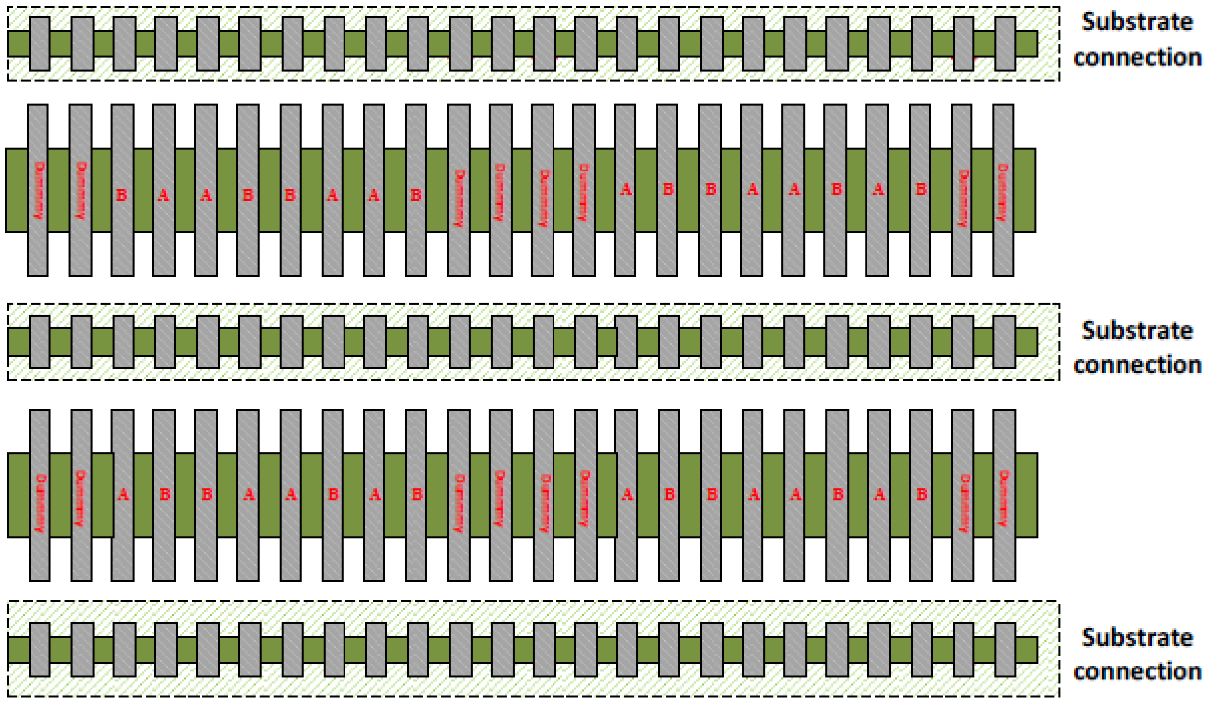

Substrate connection
Dummy Dummy B A A B B A A B Dummy Dummy Dummy Dummy A B B A A B A B Dummy Dummy
Substrate connection
Dummy Dummy A B B A A B A B Dummy Dummy Dummy Dummy A B B A A B A B Dummy Dummy
Substrate connection

Figure 37: Substrate connections

connection,

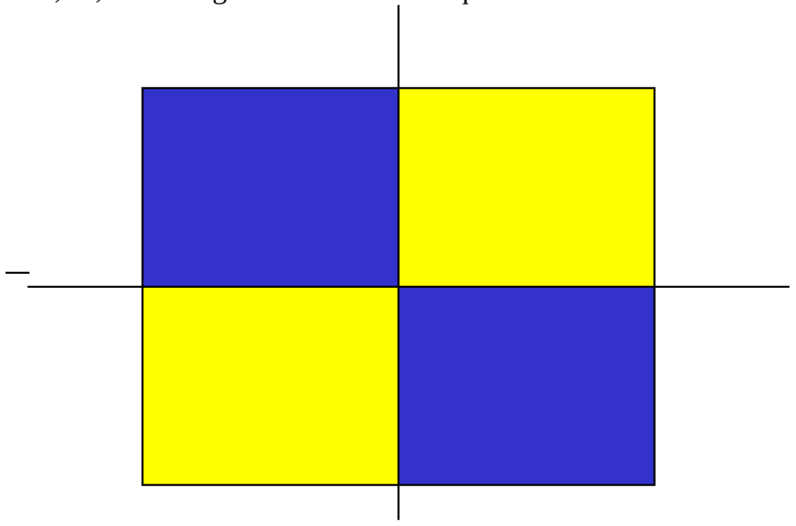
since matching between closely spaced pairs of devices is influenced primarily by local random
and systematic error factors. Matching of devices such as precise capacitors in a large capacitor-
DAC (C-DAC) array can be affected by both local and systematic long-distance effects from
lithographic patterning 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 previous figure 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.

4.3. First version of the algorithm

a- Algorithm's idea

The main idea behind this algorithm is the concept of the Quarter Cell. This concept means to divide the pattern into 4 almost identical parts. Generally, these 4 parts are hardly the same, so, we arrange them in the cross quad method.



This requires the device to be divided into units that are divisible by 4, which is the best case scenario. However, certain handling was considered for the exceptions. Exceptions are divided into 2 groups, odd numbers and even numbers not divisible by 4.

Odd numbers are a great threat to matching and symmetry because they mean one unit will remain at the end with nothing to balance the matching with. In the first version of the algorithm odd numbers were handled by creating a set of ones that contains all the remainders of the devices with odd number of units. In the current version, it is handled differently.

However, even numbers not divisible by 4 are easier to handle. A set of twos was created to contain the remainder units that can't be put in the set of 4. Which will be 2 units for each device, one unit can be put in each half in order to balance the matching and create identical halves.

Exceptions (units that are not divisible by 4) result that the quarters are not exactly identical. However, the algorithm aims at making 2 halves that are exactly identical while any device with odd number of units the remaining single unit will be placed on the center if there is any so as not to ruin the matching and symmetry.
Example:
A=7 → 4 units in the set of 4s, 2 units in the set of 2s, 1 unit in the set of ones
B=6 → 4 units in the set of 4s, 2 units in the set of 2s
C=16 → 16 unit in the set of 4
D=1 → one unit in the set of ones
E=2 →2 units in the set of 2's
F=4 →4 units in the set of 4's
Placement is done on 4 steps:
First, a quarter is created using the devices in the set of 4s, each section of the device is divided by 4 then put in the quarter. For further explanation; back to our example, quarter will consist of 1 unit from device A, 1 unit from device B, 4 units of device C and one unit from device F.
Set of 4's: A→4/4=1, B→4/4=1, C→16/4=4, F→4/4=1

Table 5: Quarter cell of first version of the tool's algorithm

C	**C**	**F**	
C	**C**	**B**	**A**

Secondly, the half is created. This happens on 2 steps, first, 2 quarters are concatenated together (a quarter and its mirror).Second, if the set of 2's is non-empty, there must be empty spots in the created half, which are filled with units of the devices in the set of 2. Considering our example, one unit of A, one unit of B and one unit E will be placed in the half.
Set of 2's: A,B,E

Table 6: Half-cell of the first version of tool's algorithm

C	C	F	**A**
C	C	B	A
C	C	B	A
C	C	F	**B**

Thirdly, in case number of rows or columns or both is odd, this means a center row or column exists, this center row or column is filled with the set of ones so as not to ruin the symmetry of the half or quarter. Also, if any units from other sets remain, they are put on this center. Considering our example, set of ones consists of 1 unit of A and one unit of D, while on 2 units from E remain from set of 2's

E
A
D
E

Fourth and last step is integration of the pattern; the half is concatenated with the center if it exists, then the result is concatenated with a mirrored version of the half.

Table 7: total array for first version of tool's algorithm

C	C	F	A	E	A	F	C	C
C	C	B	A	A	A	B	C	C
C	C	B	A	D	A	B	C	C
C	C	F	B	E	B	F	C	C

b- Challenges and Considerations:

Certain Challenges arose after implementing this first version of the algorithm:

1- Diode Connected Device

Diode connected device is the reference device in current mirrors; it must be placed in the center so its centroid coincides with the centroid of the rest of the devices. This had to be considered in the algorithm, so while placing the quarter or half, the diode connected device has a priority to ensure that it is in the center of the pattern, and then the rest of the devices are placed.

C	C	F	
C	C	B	A

A: is the diode connected device; it is placed first in the lower right corner of the quarter so that it is in the center

2- Choosing the pair

First step in creating a pattern is getting the total number of units then getting all the factors of this number to get all the possible combinations of rows and columns, which we will call list of pairs all through this thesis. Choosing the right pair is a

critical step, because this choice must be consistent with the floorplan of the whole design which this current mirror is a part of. Also, its aspect ratio must not be very extreme to ensure routablilty above the current mirror.
Last but not least, in the version of algorithm that considers a set of ones, if a set of ones exist, we must choose a pair with an odd number of rows or columns to have a center that holds the units inside the set of ones.

3- Total is a prime number

While getting the factors of the total, if the total is prime, the list of pairs will contain only 2 entries with a very bad aspect ratio (e.g total=23, list of pairs=1x23 or 23x1).
A solution for this problem was discussed was to add a dummy unit so that the total has more factors and thus list of pairs has more options to choose from.

Example 1: Total=11
List of Pairs:

Before		**After**	
Total=11		**Total=11+1=12**	
Rows	**Columns**	**Rows**	**Columns**
1	11	**1**	12
11	1	**3**	4
		2	6
		12	1
		4	3
		6	2

Example 2: Total=17
List of Pairs:

Before		**After**	
Total=17		**Total=17+1=18**	
Rows	**Columns**	**Rows**	**Columns**
1	17	**1**	18
17	1	**3**	6
		2	9
		18	1
		6	3
		9	2

Example 3: Total=23
List of Pairs:

Before		**After**	
Total=23		**Total=23+1=24**	
Rows	**Columns**	**Rows**	**Columns**
1	23	**1**	24
23	1	**6**	4
		2	12
		12	1
		4	6
		12	2

4- <u>A center doesn't exist</u>

One of the main challenges to the first version of the algorithm is that it hits a dead end if the set of ones is non-empty and a center cannot be found (odd number of rows or columns) or if the number of elements in the set of ones is greater than the size of the center.

A solution for this problem was suggested was that to change the number of units (doubling the number of units) by dividing the width of all devices. However, this is not always a valid solution, as we might hit the minimum width of the technology.

5- <u>Set of ones is too large</u>

One of the reasons the set of one was discarded, is that a center containing different devices will not harm the matching only if it contains few different devices; that are somehow near the center. However, if a long center containing multiple different devices all along its length, then it will harm the matching and shift the centroids of devices. Therefore it is recommended to do so.

6- <u>Source Shared or not?</u>

Source Sharing is a common practice among layout engineers, it cuts down the occupied area greatly and decreases parasitics. To do source sharing, 2 fingers of the same device must be present next to each other. Therefore devices can't be put randomly. To ensure that this id applied in the pattern the algorithm divides all the units by 2, places the pattern, then doubles all letters.

4.4. Current Algorithm

i- Before Choosing the Pair

A. Inputs

1) Source Shared: if =1, so we will match the devices 2 by 2

Example:

A=4, B=8, C=8

Source Shared=1

Table 8: Current version of tool's algorithm final matching pattern for source shared devices

C	C	**B**	**B**
B	**B**	C	C
C	C	**B**	**B**
A	**A**	A	A
C	C	**B**	**B**

Source Shared=0

Table 9: Current version of tool's algorithm final matching pattern for source shared devices

C	**B**	**B**	**C**
B	**A**	**A**	**B**
C	**B**	**C**	**B**
B	**A**	**A**	**B**
C	**B**	**B**	**C**

2) Width Divide: if=1, if any odds number of units is found it will double the units by dividing the width of the unit by 2

If=0, so adding dummy to the odds number of units found is the solution

Example:

A=7

B=5

C=3

If width divide=1

A=14

B=10

C=6

If width divide=0

A=7+1Dummy

B=5+1Dummy

C=3+1Dummy

3)Devices: by multiplying the number of fingers and the number of multipliers we get the total number of units for each device

Example:

A: number of fingers=5, number of multipliers=3

B: number of fingers=4, number of multipliers=2

C: number of fingers=3, number of multipliers=2

A= 15

B=8

C=6

4) Number of devices(n)

5)Technology Parameters: helps in knowing the total length and width of the pattern and the total area needed for routing, so let the user know if any excess space is needed for routability.

i. Minimum Width

ii. Minimum Metal DRC

iii. Minimum Diffusion DRC

iv. Ties Width

v. Gates Width

vi. Diffusion Length

vii. Device Width

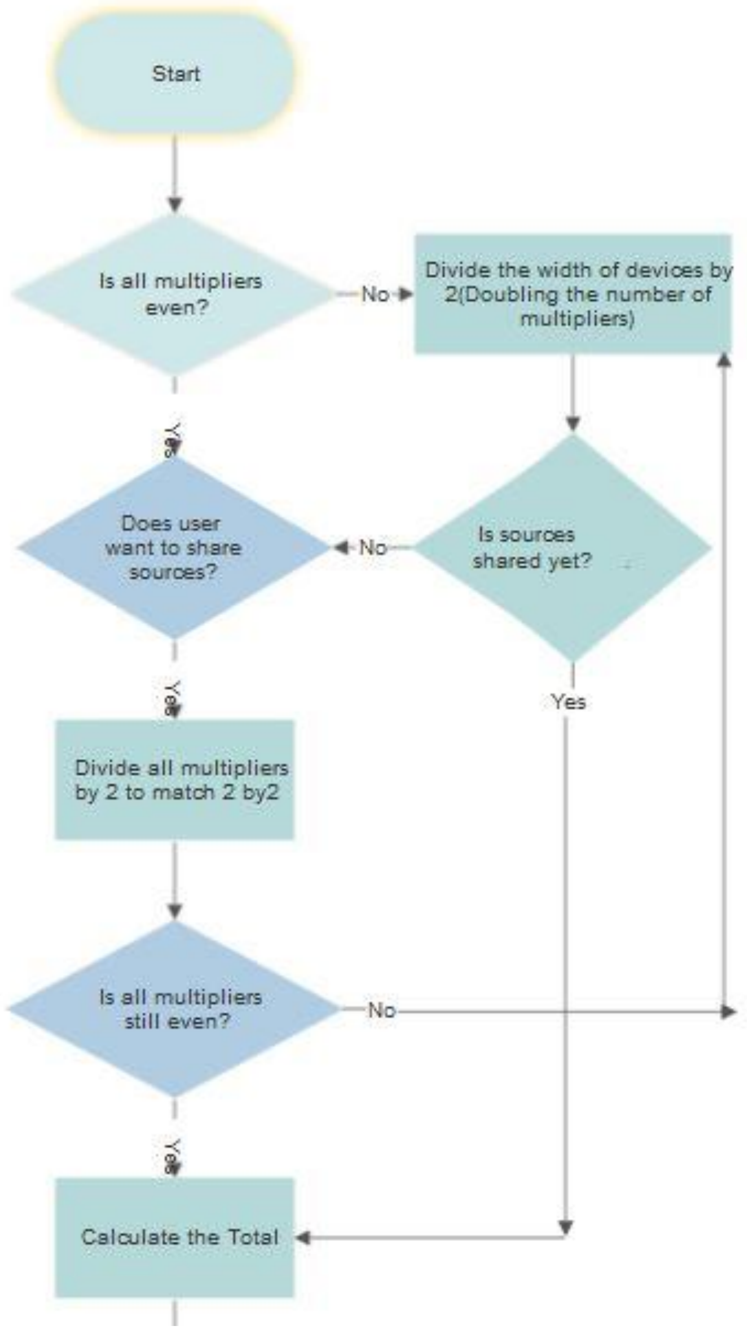
viii. Device Length

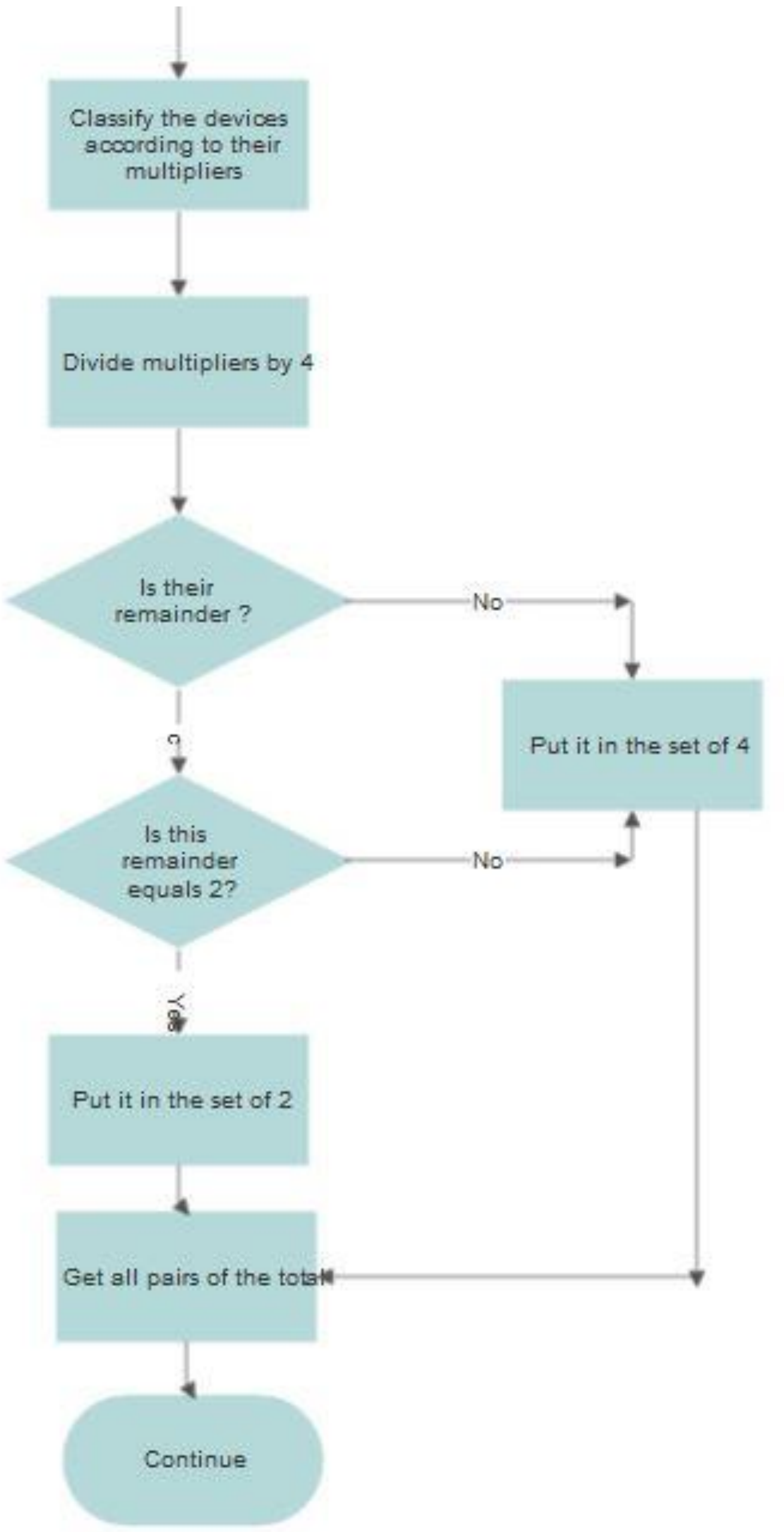
B. Outputs:

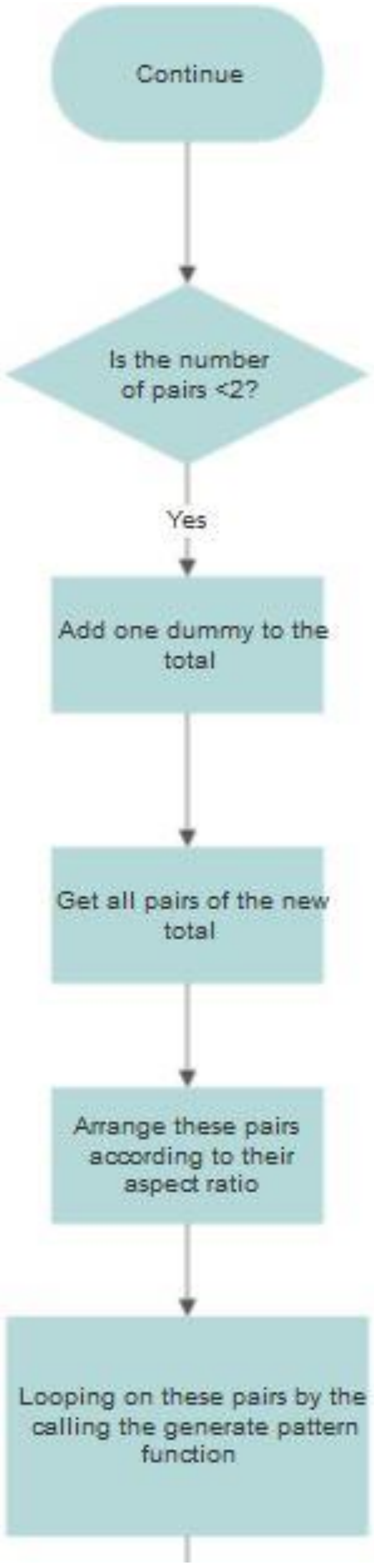
List of all combinations of rows and columns, each linked with the total length and width and the excess space if needed. The pairs are arranged according to their aspect ratio and pairs with one row or column are discarded.

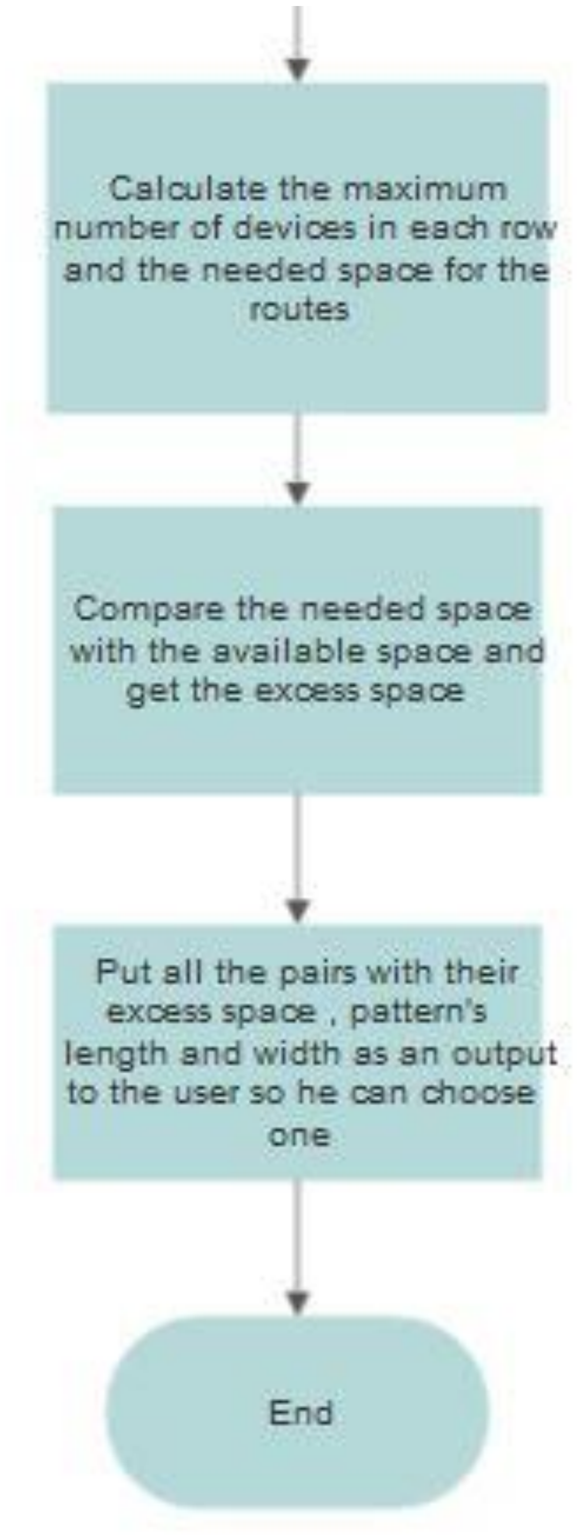
C. Algorithm

I) Solution 1

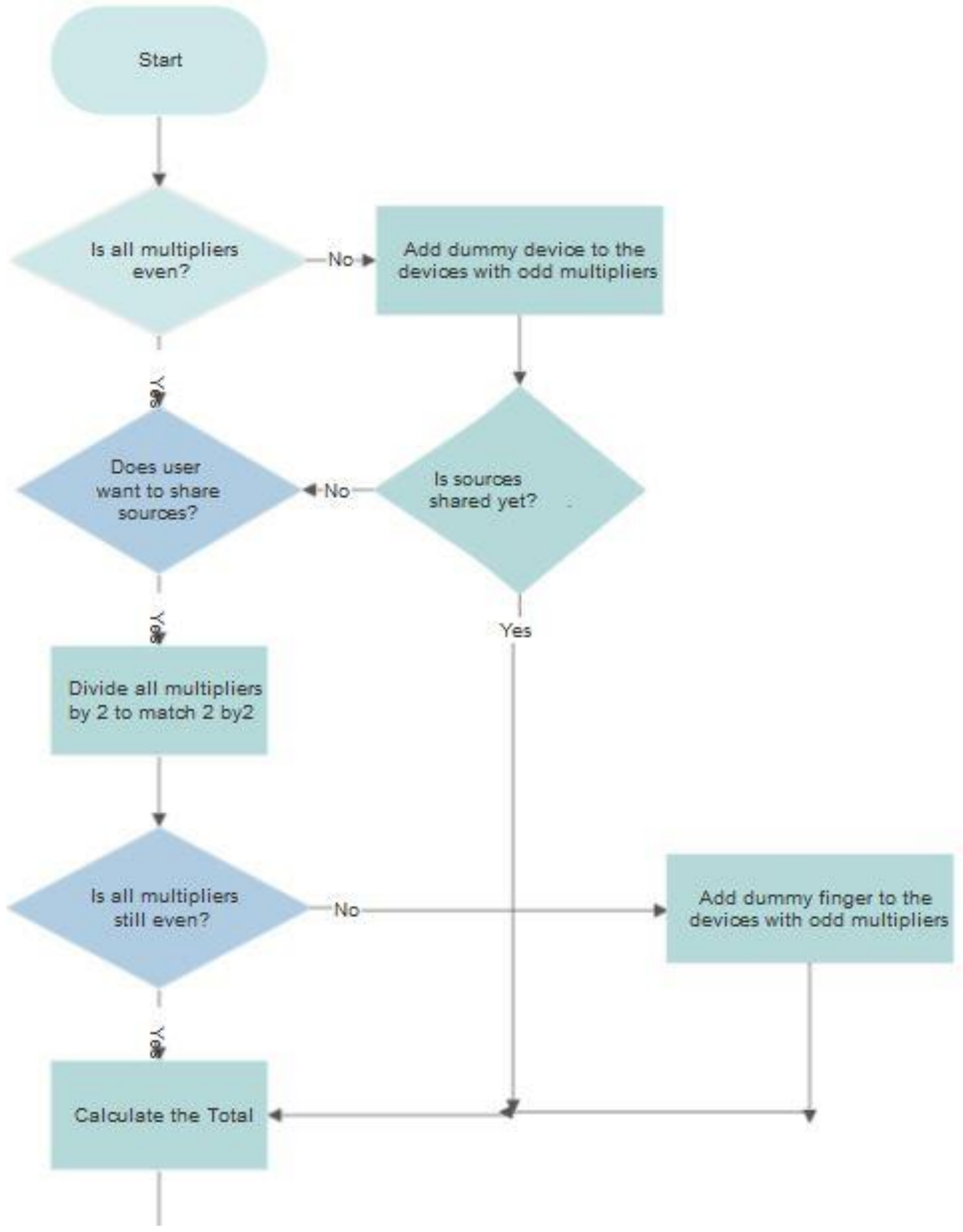

Start
Is all multipliers even?
No
Divide the width of devices by 2(Doubling the number of multipliers)
Yes
Does user want to share sources?
No
Is sources shared yet?
Yes
Divide all multipliers by 2 to match 2 by2
Is all multipliers still even?
No
Yes
Calculate the Total

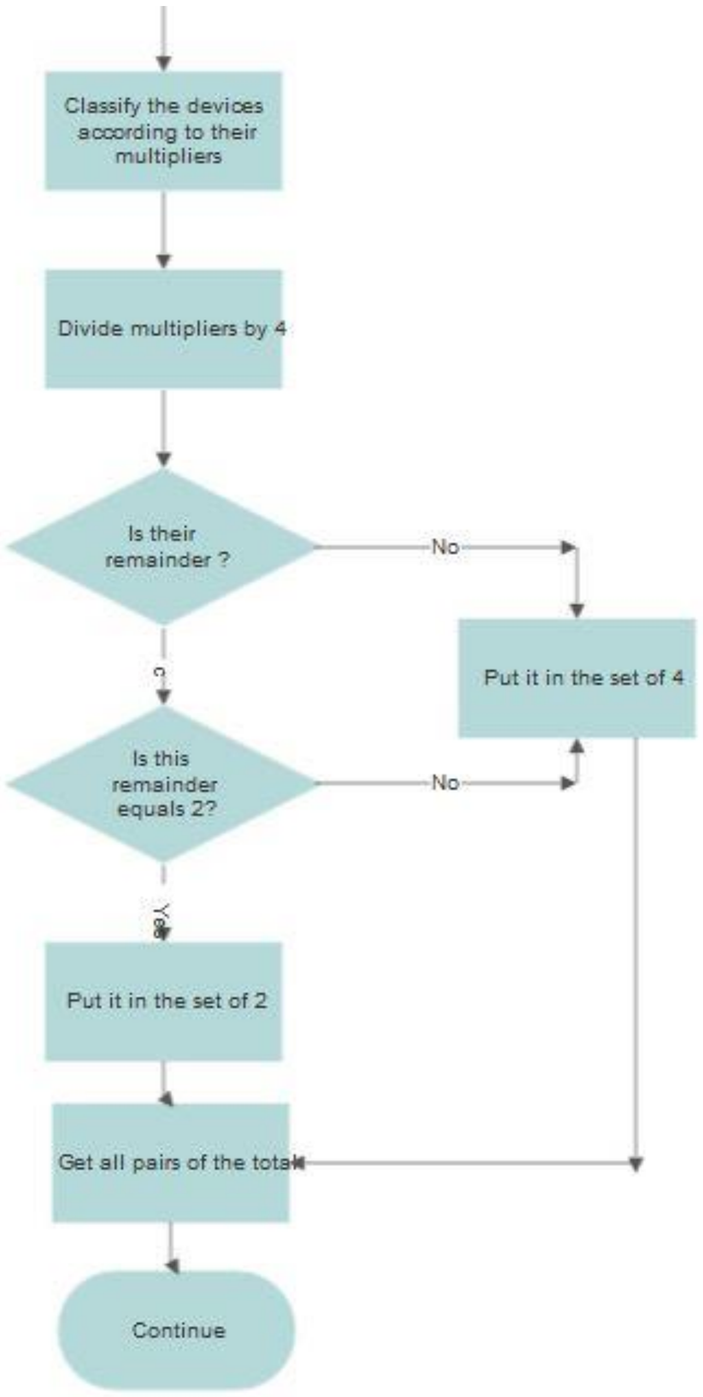

Classify the devices according to their multipliers
Divide multipliers by 4
Is their remainder ?
No
Put it in the set of 4
Is this remainder equals 2?
No
Yes
Put it in the set of 2
Get all pairs of the total
Continue

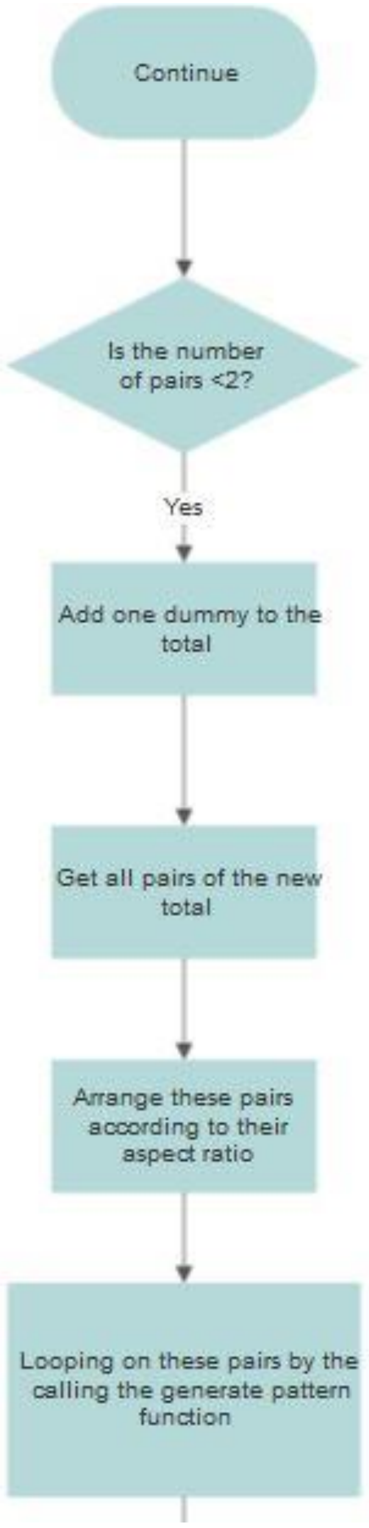

Continue
Is the number of pairs <2?
Yes
Add one dummy to the total
Get all pairs of the new total
Arrange these pairs according to their aspect ratio
Looping on these pairs by the calling the generate pattern function

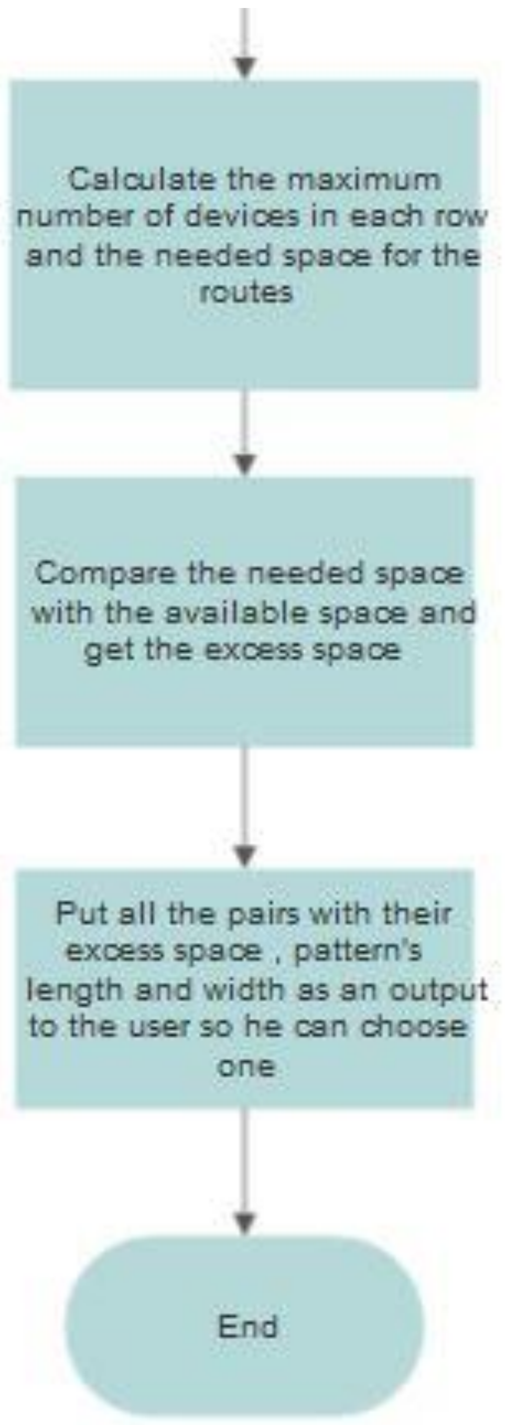

Calculate the maximum number of devices in each row and the needed space for the routes
Compare the needed space with the available space and get the excess space
Put all the pairs with their excess space , pattern's length and width as an output to the user so he can choose one
End

ii) Solution 2


Start
Is all multipliers even?
No
Add dummy device to the devices with odd multipliers
Yes
Does user want to share sources?
No
Is sources shared yet?
Yes
Yes
Divide all multipliers by 2 to match 2 by2
Is all multipliers still even?
No
Add dummy finger to the devices with odd multipliers
Yes
Calculate the Total


Classify the devices according to their multipliers
Divide multipliers by 4
Is their remainder ?
No
Put it in the set of 4
Is this remainder equals 2?
No
Yes
Put it in the set of 2
Get all pairs of the total
Continue


Continue
Is the number of pairs <2?
Yes
Add one dummy to the total
Get all pairs of the new total
Arrange these pairs according to their aspect ratio
Looping on these pairs by the calling the generate pattern function


Calculate the maximum number of devices in each row and the needed space for the routes
Compare the needed space with the available space and get the excess space
Put all the pairs with their excess space , pattern's length and width as an output to the user so he can choose one
End

ii- After Choosing the Pair

A. Inputs

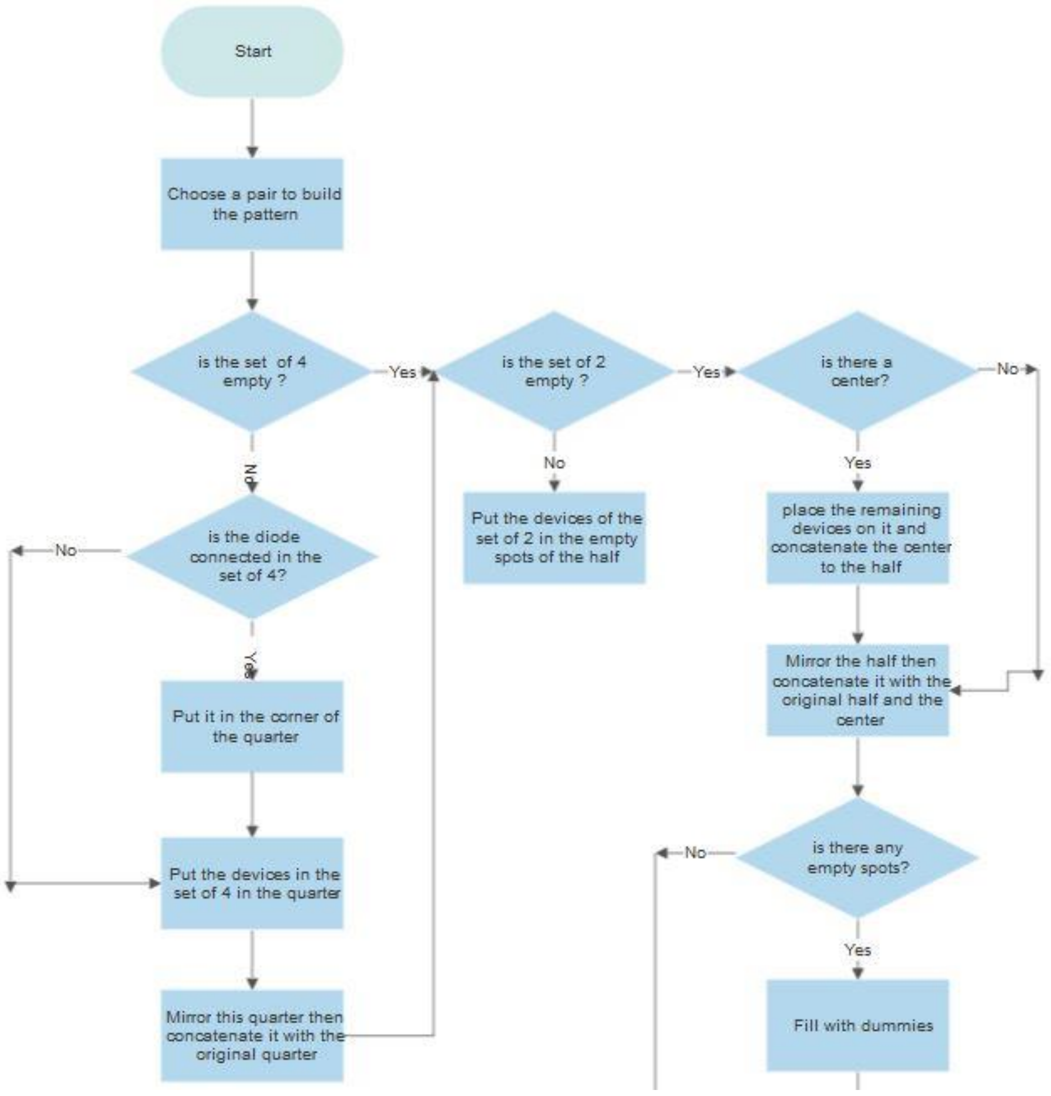
Same as the Before Choosing the Pair added to them the pair chosen (number of rows and columns) to build the pattern.

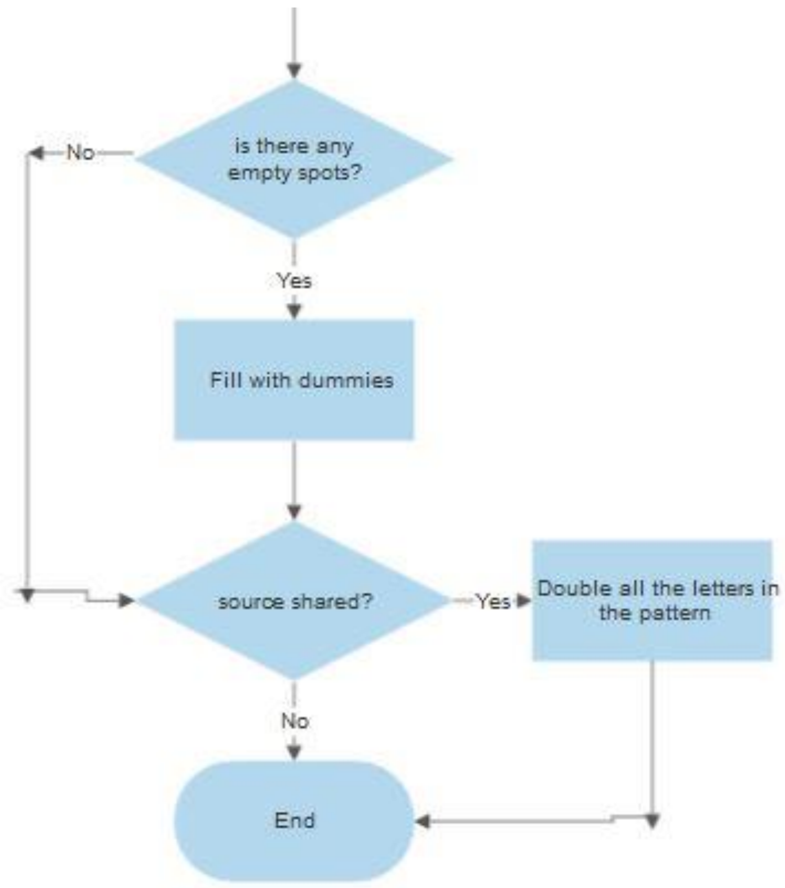
B. Outputs

The common centroid matching pattern

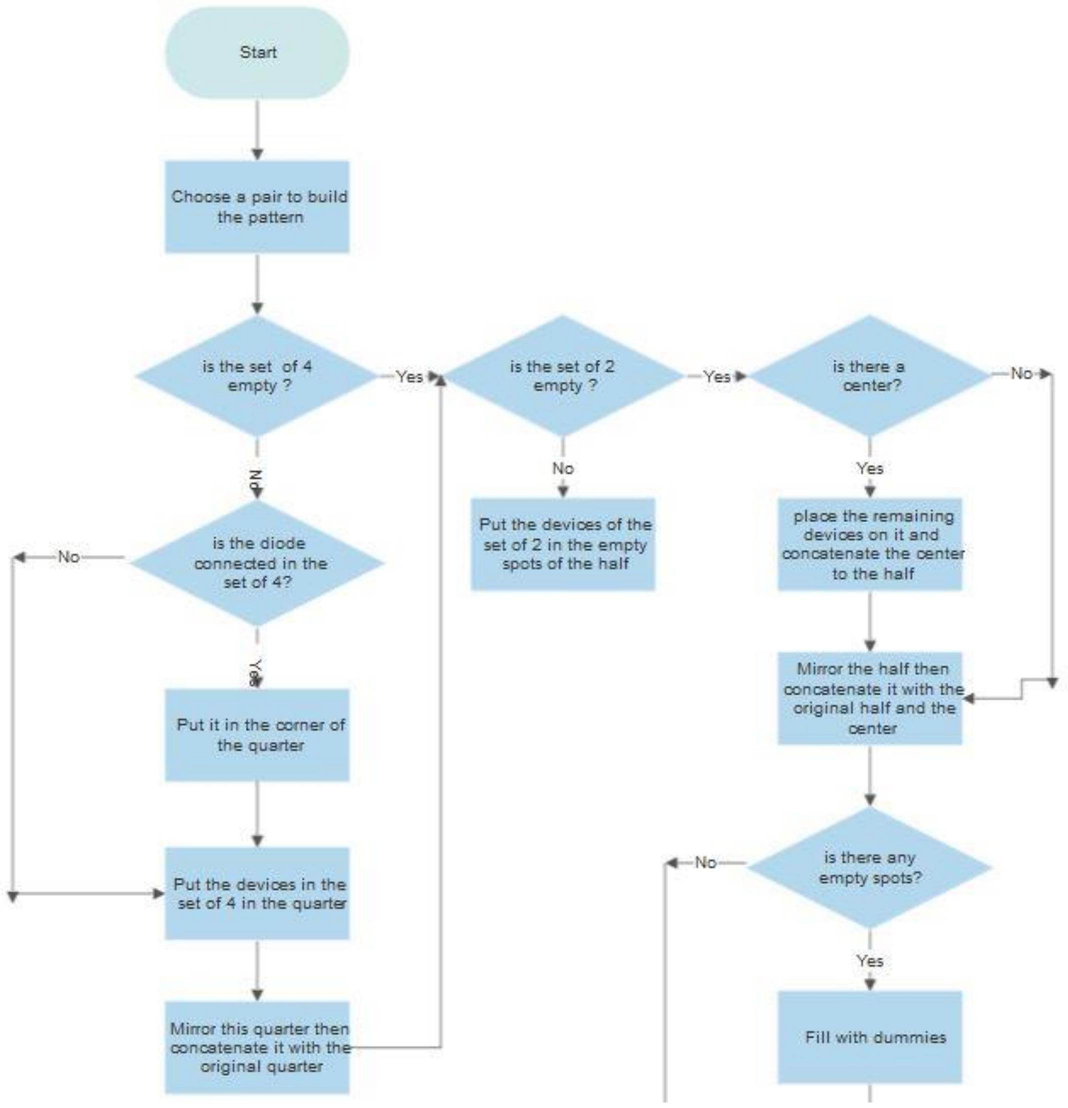
C. Algorithm

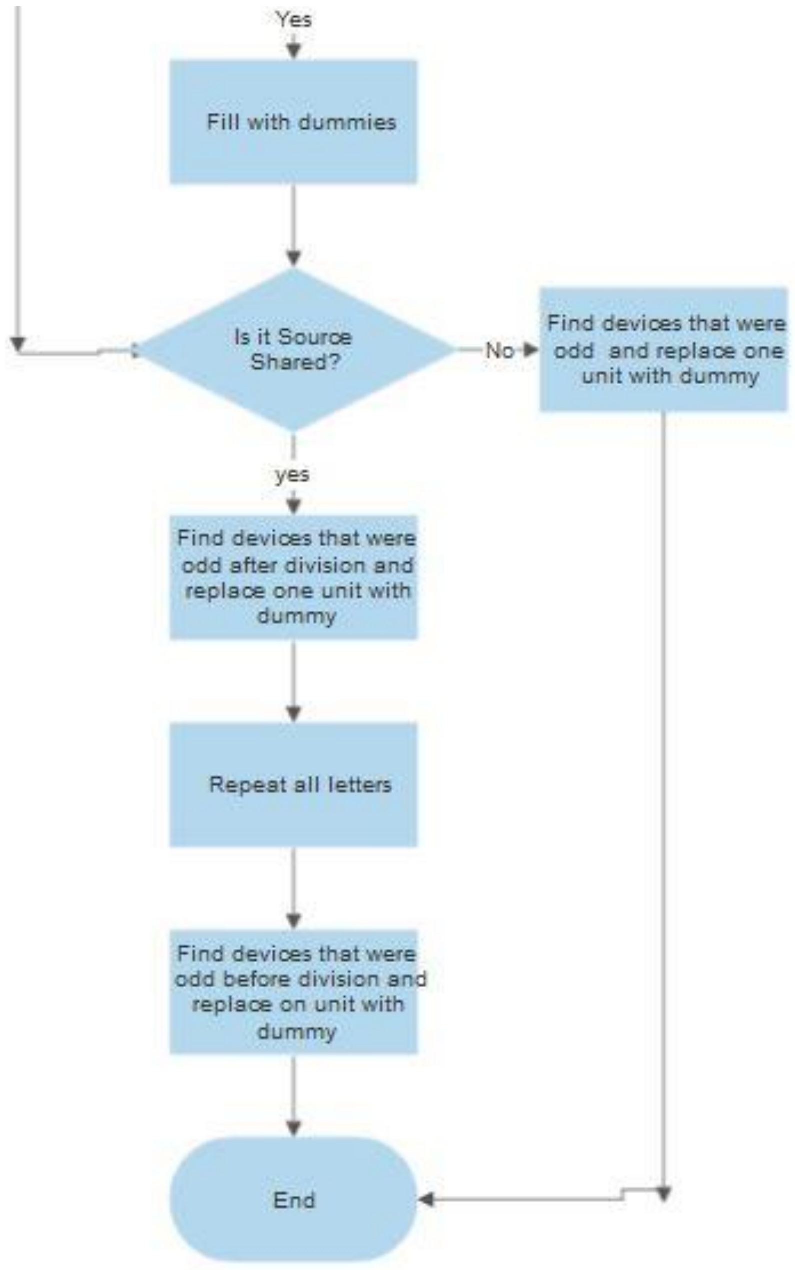
i)Solution 1


Start
Choose a pair to build the pattern
is the set of 4 empty ?
Yes
No
is the diode connected in the set of 4?
No
Yes
Put it in the corner of the quarter
Put the devices in the set of 4 in the quarter
Mirror this quarter then concatenate it with the original quarter
is the set of 2 empty ?
Yes
No
Put the devices of the set of 2 in the empty spots of the half
is there a center?
No
Yes
place the remaining devices on it and concatenate the center to the half
Mirror the half then concatenate it with the original half and the center
is there any empty spots?
No
Yes
Fill with dummies


is there any empty spots?
No
Yes
Fill with dummies
source shared?
Yes
Double all the letters in the pattern
No
End

ii)solution 2


Start
Choose a pair to build the pattern
is the set of 4 empty ?
Yes
No
is the diode connected in the set of 4?
No
Yes
Put it in the corner of the quarter
Put the devices in the set of 4 in the quarter
Mirror this quarter then concatenate it with the original quarter
is the set of 2 empty ?
Yes
No
Put the devices of the set of 2 in the empty spots of the half
is there a center?
No
Yes
place the remaining devices on it and concatenate the center to the half
Mirror the half then concatenate it with the original half and the center
is there any empty spots?
No
Yes
Fill with dummies


Yes
Fill with dummies
Is it Source Shared?
No
Find devices that were odd and replace one unit with dummy
yes
Find devices that were odd after division and replace one unit with dummy
Repeat all letters
Find devices that were odd before division and replace on unit with dummy
End

Chapter 5.
Automatic Routing of Current Mirrors

5.1. Opening thoughts on routing of analog devices.

Routing, which determines the interconnections among devices, is one of the most significant stages in the layout automation flow. In the analog layout tools described in chapter 2, the area routers are widely applied, where the routability is largely dependent on the routing sequence of the nets. Because of the separation of the placement and the routing phases, the conventional device-level placement algorithms typically apply routing area estimation to allocate interconnection space. However, failure to allocate sufficient interconnect area may generate an unroutable placement solution while the overestimated routing area may result in the loss of density. As a matter of fact, accurate routing area estimation is impossible in the placement stage of the above proposed algorithms. Even worse, a rough estimation of routing area tends to mislead the optimization search during the placement. Although LAYLA router (described in chapter 2) proposed a dynamic routing area estimation technique, which is claimed to have a significant improvement over those static approaches, 10% to 15% area redundancy still exists when comparing their results with the layouts done manually. The analysis of the dense expert-handcrafted analog layouts has revealed that the interconnection among analog modules occupies much less area than the area taken by the modules themselves. Since our module normally consists of several closely related devices, and the interconnections among devices have been completed inside the module, the number of inter-module routes is less than that of the interconnections among single-devices. In addition, the inter-module routes occupy much smaller area than those of the modules. As a result, the routing area estimation is excluded in the placement stage of ALADIN, and two routing phases, including the global routing and the detailed routing, are employed. The global routing is integrated into the placement procedure to improve the accuracy of the routing path estimation. The compaction-based constructive detailed routing completes the final layout based on the output from the placement procedure.

In this previous work, the significant drawbacks are:

1) Failure to allocate sufficient interconnect area, this failure may result in one of two cases, either underestimated routing area, which leads to unroutable placement, the other case is the overestimated routing area, which leads to a huge waste in silicon or, at least, results

in redundant routes. The wasted silicon impacts in more chip cost, which will affect the total mass production cost.

2) No consideration for symmetry requirements, or for the signal flow. sometimes the signal flow has been meticulously planned and detailed by the circuit designer for a very important reason, like the following figure:

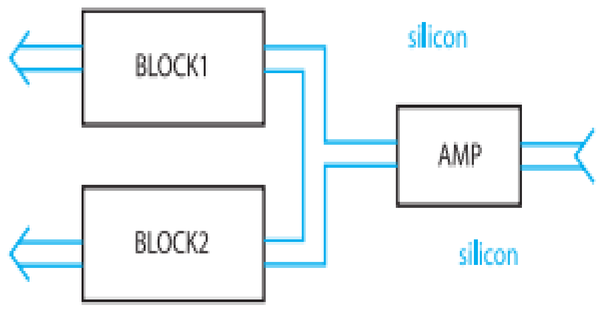


Figure 38: Sometimes you are given fixed wiring layout.

In the figure above, the symmetry is the most important element of the circuitry. We have an amplifier that feeds two blocks, and there is a need to have a horizontal line of symmetry through the whole thing. The physical designer is told the blocks cannot be moved.

In this particular case, the physical designer notice there is some wasted silicon above and below the amplifier. he might complain, “But, I am trying to make my chip as small as I can, and if I can move the amplifier north or south then I can make the chip much smaller.”

To which he might hear, “But then you would lose the symmetry. The circuit relies on the signal from the amplifier arriving at the two output blocks at exactly the same time. If we don’t get the signals exactly synchronized the circuit will not work. The symmetry is a requirement.” So, again, here is another example where the mask design is influenced by the function of the chip.

In the coming section we are going to present our router, and how it found an innovated solution for the previous drawbacks.

5.2. ACML router flow

In this section we are going to introduce our innovated Automated Current Mirror Layout router (ACML router), which was built from scratch, based upon TCL scripting, which is the scripting language executed by Synopsys custom compiler.

The main breakthrough of the ACML router, that it doesn’t start of thinking for a the best routing solution after having the transistors already placed, like the traditional tools, it also doesn’t perform simultaneous placement and routing algorithm, in its direct meaning, although, the tool treats routing as “placement of interconnections.” It starts this stage of placement after having a complete placement of the CM array, fulfilling the matching requirements (the matching pattern is passed to the placer by the C++ code.), and having the dummies at the two sides of the array. The ACML router doesn’t, afterwards search for the best solution, it implements a solution based on imitating the manual methods which were introduced to us at the Analog layout training by Silicon Vision Company.

In other words, the ACML router doesn’t search for a solution, it knows the best one, it implements it, and it unbelievably saves hours and hours of hard, boring, and systematic work, a manual routing for a big size current mirror may take around 2-3 hours on average, this 120-180 minutes reduce to 1 minute by the help of this router.

This imitation based solution will be introduced step by step in the following sections, according to the following flowchart:

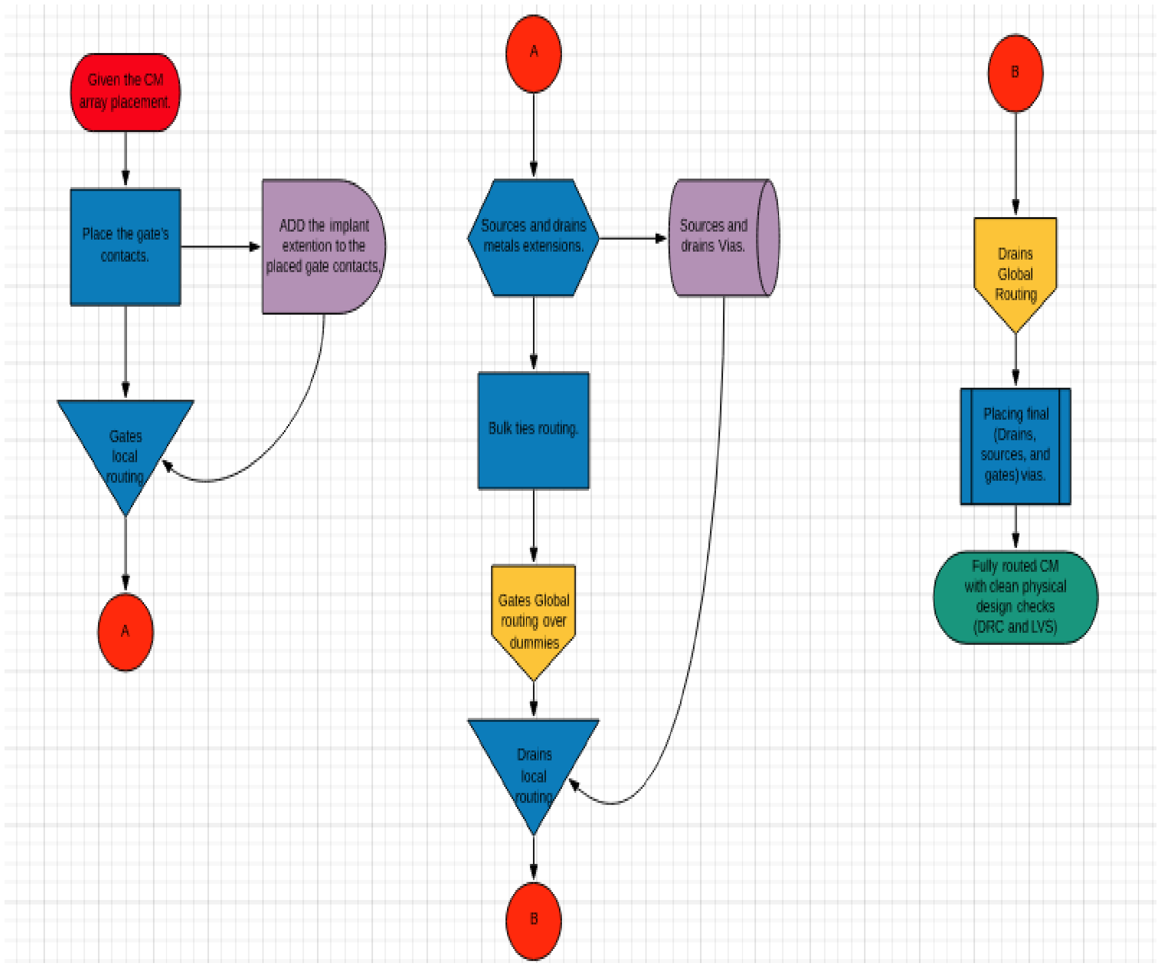

Given the CM array placement.
Place the gate's contacts.
ADD the implant extention to the placed gate contacts.
Gates local routing
A
A
Sources and drains metals extensions.
Sources and drains Vias.
Bulk ties routing.
Gates Global routing over dummies
Drains local routing
B
B
Drains Global Routing
Placing final (Drains, sources, and gates) vias.
Fully routed CM with clean physical design checks (DRC and LVS)

Figure 39: ACML router flowchart

.

5.3. Placing the gates contacts

The tool we are working on (synopsis tool) can place the transistor with all the gates connected together by polysilicon but from the training we took at silicon vision under the supervision of Eng. Fady we know that it is not good to connect between gates by poly as this puts a lot of resistance and capacitance on the gate as polysilicon has high resistance and

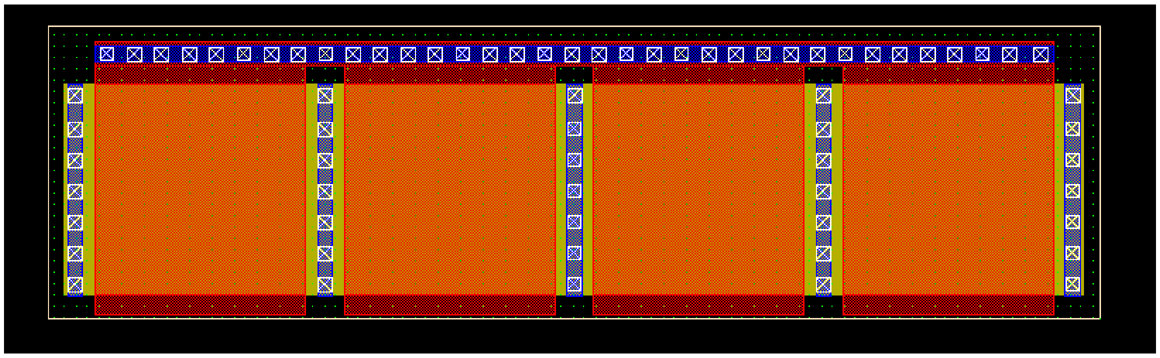


Figure 40: The default gate connections implemented in the p-cell, with undesired U-shaped poly routing.

capacitance per unit area and the following figure shows that default connected made by the tool

So to avoid this disadvantage and to make our product aligned with the industry we place the transistor without the default gate connection and we will make this connection to each finger alone then connect all the gates together with metal 1 so now the poly is not used to connect all gates and the following figure shows the placement of the fingers without gates contacts

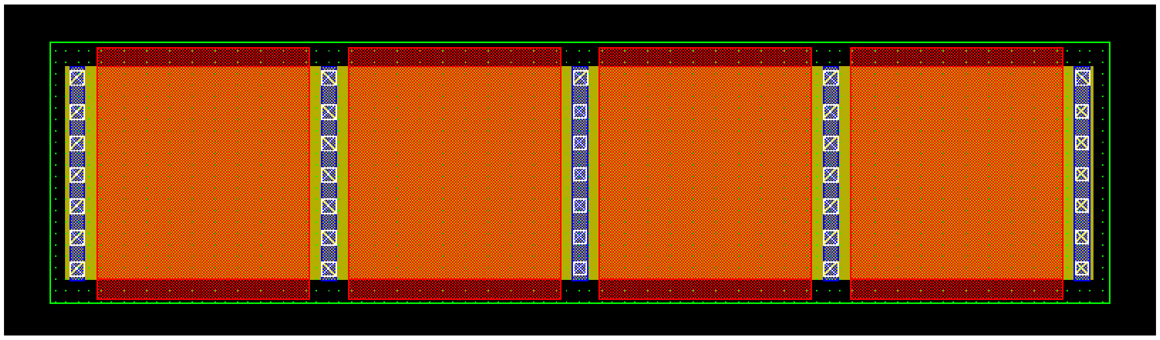


Figure 41: Transistors with gate contacts option disabled

And now we will explain how this function work in order to reach our goal.

Firstly ,we made an virtual line that intersect the placement of transistors vertically to know the number of rows and made a for loop to loop around this number and inside the loop we made another virtual line inside the loop to intersect the poly of the to know the coordinates of each finger and by using this coordinates we will put extra poly above each finger and above it we will put a gate contact from the poly to metal 1 in order to connect all the gates of the fingers together and the following figure shows the fingers and above each one the extra poly and the

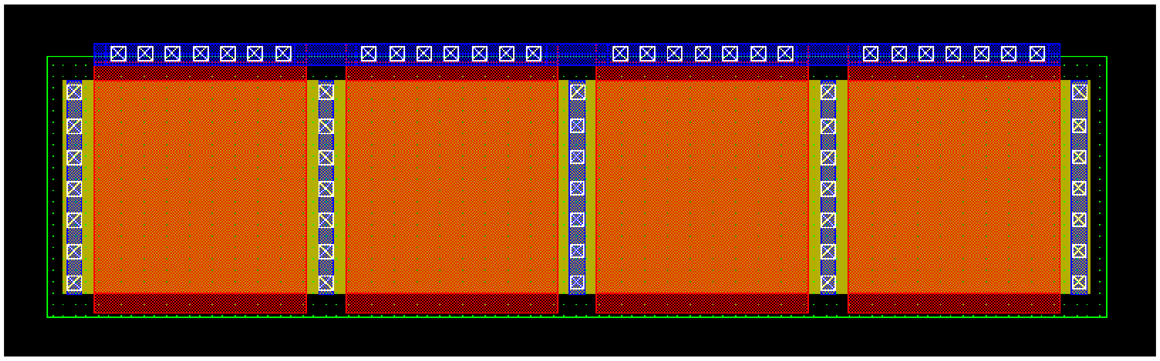


Figure 42: Gates connections with metal 1 instead of poly

contact.

5.4. The implant extension

In order to follow the design rules and ensure that the layout is manufactural the router must extend the implant over the gates with the specified design rule of the specific process design kit (PDK).

This design rule varies from one kit to another so the tool reads this extension value from the PDK and extend the implant to ensure proper etching effects over all the gates.

After reading this design rule the tool defines the boundary box of this extension from the lower left point to the upper right point and starts to draw this extension.

These points are determined by drawing a virtual line that intersects with the previously placed contacts with the help of TCL function (get intersect) this will specify the lower left point of the box then the upper right point of these box is determined by the DRC value.

After that the tool loops over each row to draw the extension of the implant and to ensure the best layout which is 100% manufactural.

In the following figures we see the layout after extending the implant with the proper DRC value

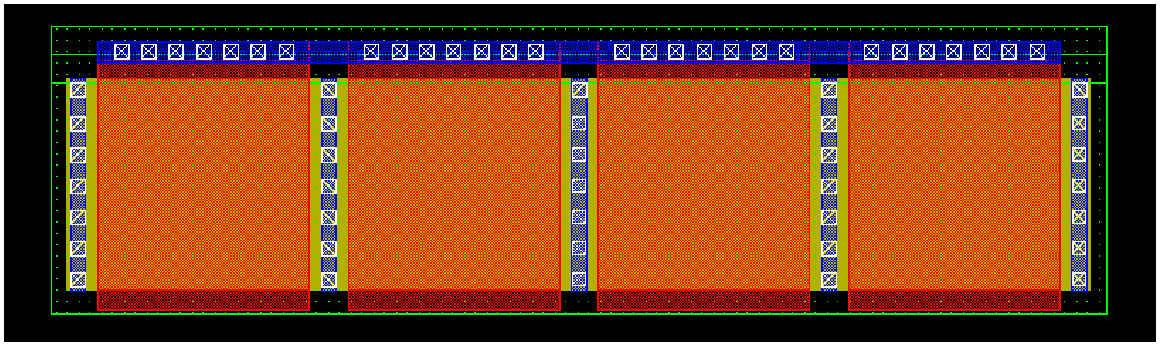


Figure 43: NMOS transistors with the NIMP extension

5.5. Sources and drains metal extension and putting vias

In this part the tool first tries to make it easy to connect all the sources and drains of the

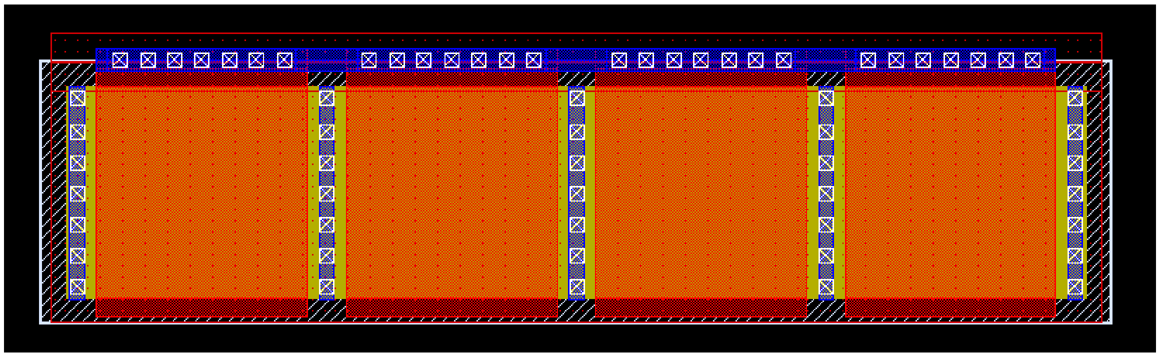


Figure 44: PMOS transistors with PIMP extension

devices.

Our approach in this thesis and in the implemented tool is to connect all sources together in case of source shared units, this approach depends on connecting all the sources together with thick metal to ensure that it can handle the current flowing through it and also pass the metal filling design rules of the kit.

In case non source sharing each source of each device is connected individually.

After that all the sources are connected together to the supply rail either VDD or VSS by implementing a connection called a “ Star Connection “ : in this connection a very thick metal rail is drawn for the supply VDD or VSS then all the rails that will connect to supply will connect individually .

In order to be able to do all of that a metal extensions over all the sources with metal 2 , after that VIAS from metal 1 to metal 2 are placed over each source.

Identifying the sources is done by drawing an intersection line that intersect with source and drain diffusions and by looping over each row we can easily identify the location of all sources in each row so we can draw the metal extensions and place the VIAS at the exact locations.

In order to draw these metal extensions a TCL command “db::createpath” is used , this command takes the start and end points as its attributes to draw the defined path, Then a command “placevia” is used to place the vias mentioned above.

After finishing the source connections here comes the turn of drain connections, the drains of each device is connected individually to a pin through vertical metals.

In order to be able to do this metal extensions also must be drawn over each drain and then vias are placed with the same procedure mentioned above.

The connections of the drains are different from the sources which were all connected together to the supply which is easier than that, but for the drains of each device will be connected separately to the outside world so it takes another way.

The following figure shows the CM array after Sources/Drains extensions

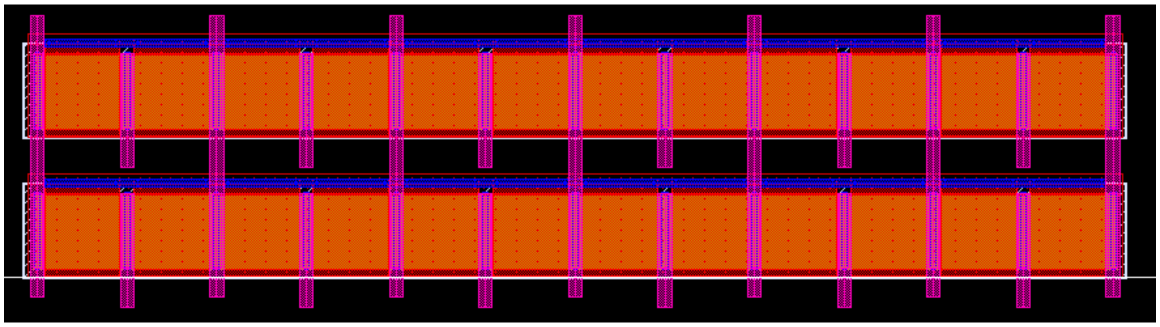


Figure 45: CM array sources and drains extensions

5.6. Bulk connections

. In this part of the thesis we are going to talk about the bulk ties. Connecting the bulk ties to a very strong supply with a very strong connection is one of the most important things that a good physical designer must do. In this way we ensure the PN junction between the active areas and the substrate will be always reverse biased and this will prevent any latch-up and current leakage from the circuit.

To ensure very strong connection the star connection approach mentioned above is used with the rails of VDD and VSS implemented by stacked layers of metals to ensure low resistivity and strong supply lines. The first thing this tool does is to determine the total bounding box of the already placed devices this returns two points: the lower left point and the upper right one , these points are latterly used to place the ties and route them.

The first thing done is to place the high tie using the points determined before (the upper right point) and the other point is determined by the minimum DRC of the tie contact and the implant enclosure around it, then the tool starts to place the implant with the defined enclosure around the contact then place the contact from substrate to metal_1 inside the implant. After that the tool loops over each row to place the contact with the same way the contact is placed above with the implant enclosure around it.

Then the lower tie is placed in the same way. To ensure very good isolation and strong connection to the supply, ties also placed from both sides to form a guard-ring around the mirror. After that all the ties are connected together using metal_1 through supply with the star connection.

The following two figures show a PMOS and an NMOS CM arrays respectively after tying their bulks to the supply voltages.

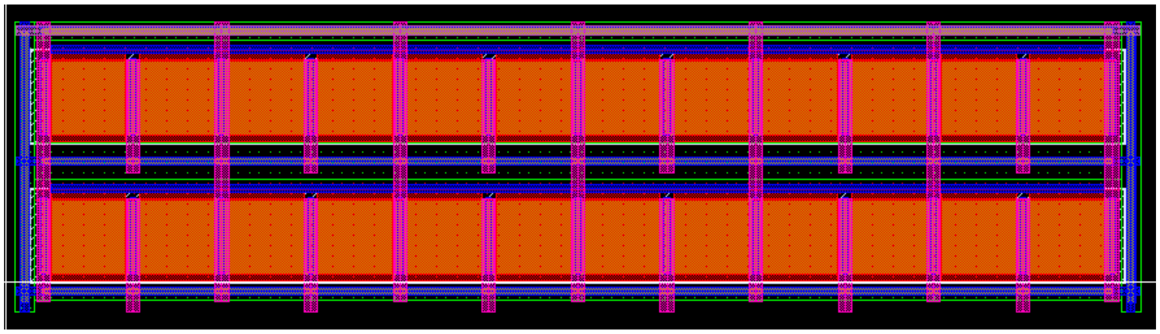


Figure 47: PMOS array with its bulk connected to VDD

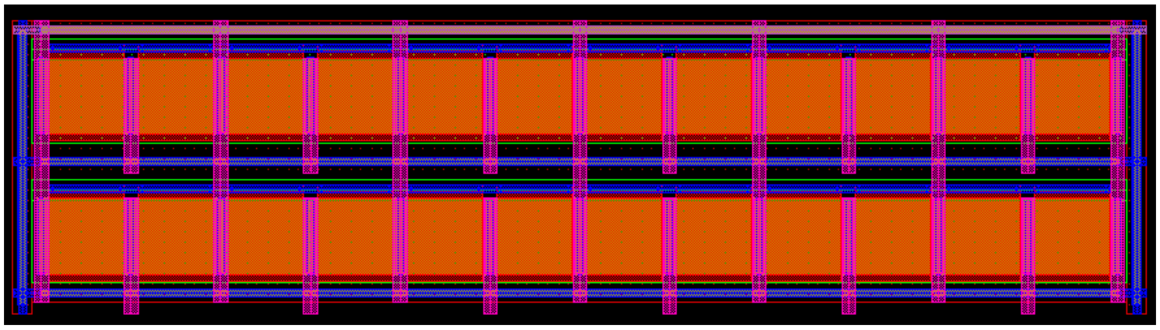


Figure 46: NMOS array with its bulk connected to VSS

5.7. Gates global routing

The tool we are working in it (synopsis tool) can place the transistor with all the gates connected together by polysilicon but from the training we took at silicon vision under the supervision of Eng. Fady we know that it is not good to connect between gates by poly as this puts a lot of resistance and capacitance on the gate as polysilicon has high resistance and capacitance per unit area. That’s why we are not using this built in function and we implemented our own function that places the gate contacts and route the gates of each row together. As we know in current mirrors all the gates of all the devices of the mirror must be connected together through a very strong connection, so we make use of the area above the dummies and connected the gates of each row together by a vertical rail that gathers all the gates of the mirror. Using metal 2. This connection is also used to gather the drains of the root device, the diode connected device, in order to decrease the number of metal 4 vertical rails required to globally connect the drains.

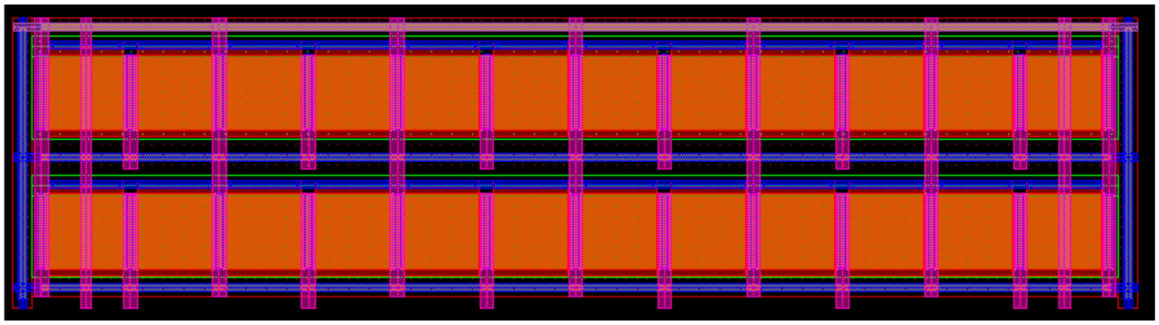


Figure 48: gates global routing

The following figure shows the CM array after performing gates global routing.

5.8. Drains local routing

After all the previous work now we need to route all the drains of the same transistor together and this will be done in two steps. Firstly, we will route drains internally by horizontal of metal 3 and secondly we will route the drains globally by vertical rails of metal 4

So in this section we will discuss the first step of connecting the drains internally by rails of metal 3 to connect the same drain from the different fingers of it.

To do this from the matching pattern we need to count number of different transistors in each row by 2 nested loops to loop around rows and columns of the pattern and we get the maximum of this numbers in order to work on it so in each row we will put this number of metal 3 routes to fulfill the needs of the rows that contain the maximum number of different transistor and in the rows that contain less number of transistors there will be extra rails that we will connect it to VSS in case of NMOS transistor and VDD in case of PMOS.

And to fulfill the current capability of the metals we made with different widths in order to prevent electromigration from damaging the routes and the width of every route is determined by the current we need it to carry

Now we can explain the algorithm of putting the horizontal routes. Firstly, we calculate the area or the channel we will put the routes on and then we add the the metals routes to be put in this area than we subtract this value from the total space so now we know the total area available for spaces then we divide this area by the number of routes minus one so we know found the single space and if this space in less than the DRC rule between we will increase the routing channel to can accommodate this number of metals

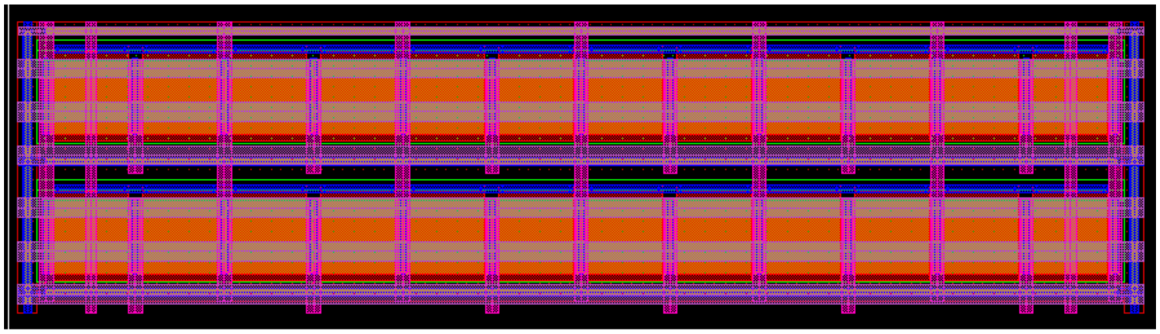


Figure 49: Drains local routing

The following figure shows the transistor with the drains horizontal routes:

5.9. Drains global routing

Now that we have established routing channels, making use of the area over the transistors, in addition to the bulk ties area, to minimize the channel area, and after the local drains rails, that routes and collects drains of parallel fingers in the same row, in this section we can start to examine each of the individual channels in more detail. Each channel has been identified as containing a certain number of signals. How can signal flow from one channel to another without a lot of jogging and heavy unnecessary metal layer changes. In other words, how can we collect all horizontal rails (gathering the drains of parallel fingers for a certain transistor in the CM) with a wide chunk of vertical metal, in a star connection?

The great advantage of this star connection, is the significant decrease in the node resistance caused by interconnecting the drains in this way, imagine the flow of current starting its journey from this vertical rail, wide enough to handle the current flowing without being exposed to electromigration phenomena, it’s then divided equally to the horizontal rails, which makes the current flowing faces a parallel connection of resistors here, at this case, the transistor node decreases by $\frac{1}{number\ of\ horizontal\ rails}$ one may imagine that at this case, the best mythology is to route those drains at the same row with much more horizontal rails in order to decrease the drain resistance, good for your resistance but how about parasitic capacitance right now? The chip may be killed this way. Another consideration is the routing channel, we cannot assign the whole channel for only one transistor.

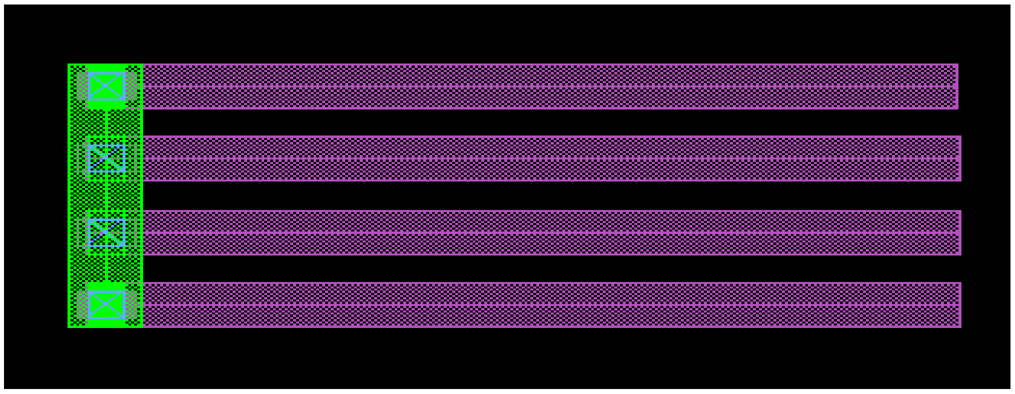


Figure 50: star connection using metal 3 and metal 4

In the previous image, the proposed star connection is declared, horizontal metal 3 rails run in parallel, then they are all gathered by vertical metal 4 rail.

The global routing of drains procedure relies on two main inputs, the local routed drains passed by the previous procedure, and the total number of different devices in the current mirror (the number of devices after parallel reduction, after exclusion of fingers and multipliers.). Which are passed from the placer ordered and numbered. For example if we have a CM that is consisted of the root diode connected device, and three other devices to whom the current is mirrored, then the input to the global drains routing procedure will be (M1 M2 M3 M4). Hence, the number of required vertical rails is calculated, which is 4 In this case, however, the router doesn’t only go to the placement and put 4 equally spaced rails above the devices, as at this case we may observe a routing mismatch.

In order to keep the routing above the transistors in a perfect matching, the router reads the total number of columns in the CM array, it places one wide metal 4 rail above each column (excluding dummy columns), after calculating the bounding box of the CM array, it sets the length of each rail to the whole length of the array, to guarantee it will cover all rows of transistors. The rails required to collect all the drains are now used in order: rail one to connect the second device M2, rail two to connect the third device M3, rail three to connect the fourth device M4, and so on. (Note that the root device is connected using vertical metal rails above the dummies, connecting them to the transistors gates as well.) The extra (and there may be extra) vertical rails are then connected to the supply voltage, not to be left floating. Now one may say: there are some extra overlaps in the drains wiring that do not need to be there. Ok, that's right,

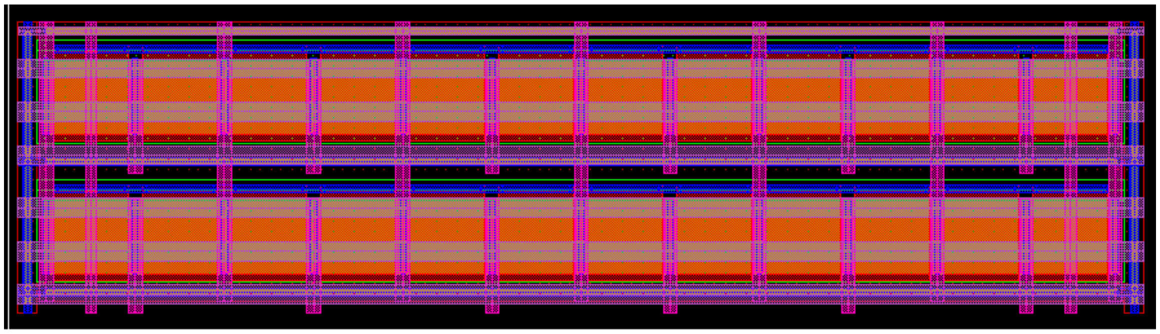


Figure 51: Before drains global routing

however, the extra overlaps help balance some of the crossover parasitics. We try to make the wiring the same length, with the same overlaps, on the same metals—everything identical. A perfect matched and symmetric current mirror isn't only about placement, routing is a concern too.

In the previous figure we can see the CM array before the placement of the global drains rails, and in the following one the procedure were called, and the vertical metal 4 rails are placed

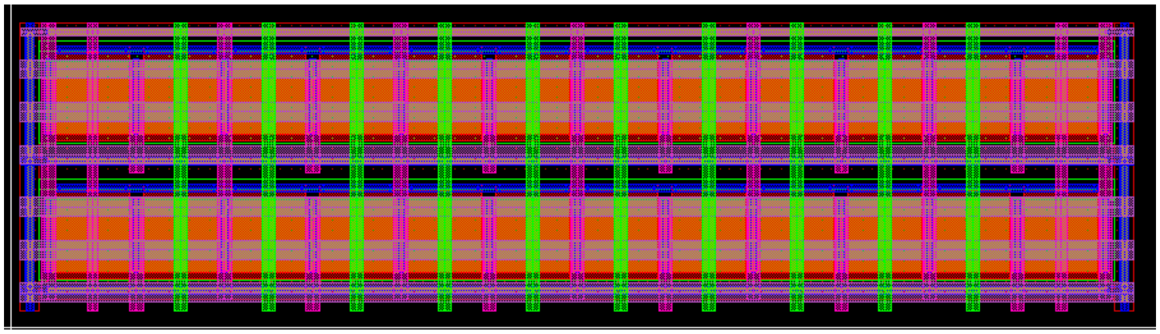


Figure 52: Drains global routes

in the way described.

The procedure, in fact doesn’t physically connect those vertical rails to the corresponding horizontal ones, it places them in order, and give them the same net name as the corresponding horizontal rails, which is also the same name of the transistor in schematic viewer (M1, M2, M3, M4). Naming the rails very carefully is the key role and the main target of this procedure, if every vertical rail is related to its corresponding transistor by the identical names, and the extra vertical rails are named after the name of supply nets, then the physical and actual connection of the drains will be easy, it will be done using connection by name methodology, which will be described in details in the following section.

5.10. Placing Metal VIAS

Good work always needs the final touch to be ready and complete, without this final touch all the good work and efforts that have been done can be gone.

As we mentioned in the previous parts of this section that the routing of the current mirror is done step by step, first the gates then sources and drains then the bulk.

So in order to complete this flow VIAS must be place to connect different metal layers to each other's..

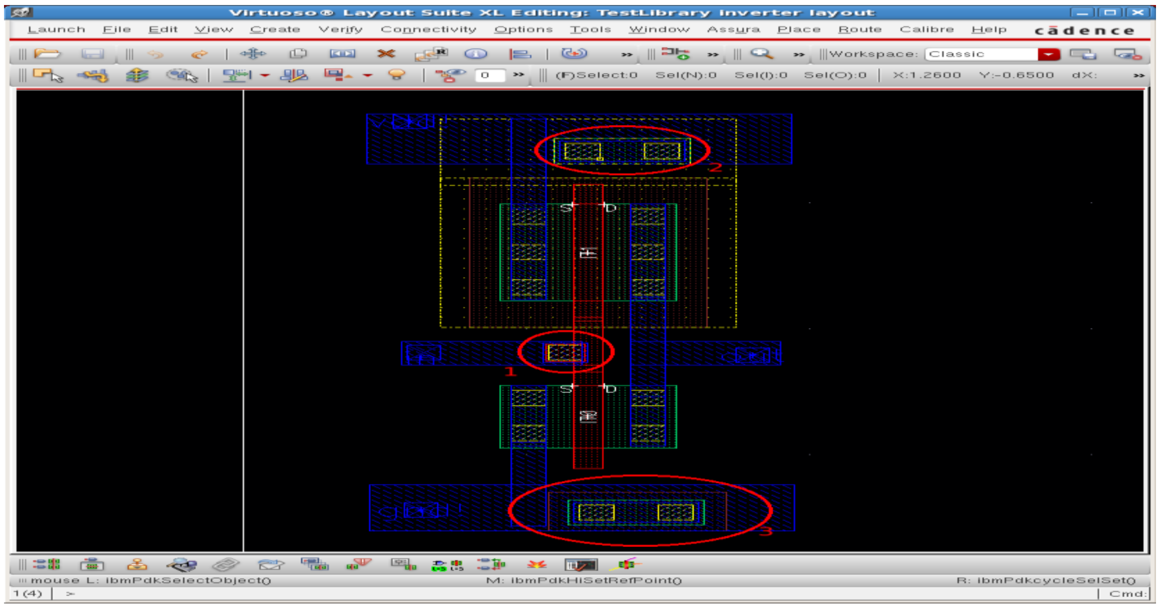

Virtuoso® Layout Suite XL Editing: TestLibrary inverter layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura Place Route Calibre Help
cadence
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:1.2600 Y:-0.6500 dX:
1
2
3
mouse L: ibmPdkSelectObject()
M: ibmPdkHiSetRefPoint()
R: ibmPdkcycleSelSet()
Cmd:

Figure 53 : VIAS in Layout view

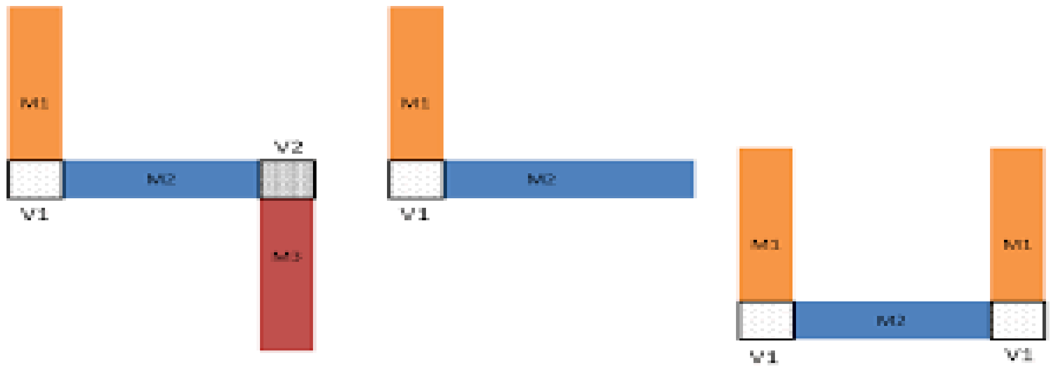


Figure 54 : VIAS between different metal layers

VIAS are the most exhausting thing that a physical design engineer does, forgetting a single VIA can cause a lot of problems and LVS errors that you can spend hours and hours searching for it if you don't have some helpful tricks in your tool that tell you where is the missing via and tell you if you missed a via or the error in something else.

Some tools just issue some LVS errors and don't specify where are those errors and what kind of errors do you have , so automating the placement of vias will save a lot of time , effort and cost.

First the tool starts to place the vias of the local interconnections (from the metal 2 extensions on each drain to the horizontal metal 3 rails over the device) to connect all the units of the same device in each row together in a single row.

It's not an easy task but we did it, first in the phase of drawing the meal extensions over each drain we give this net the name of the device so now we have each device with a metal 2 extension carrying the same name of the device.

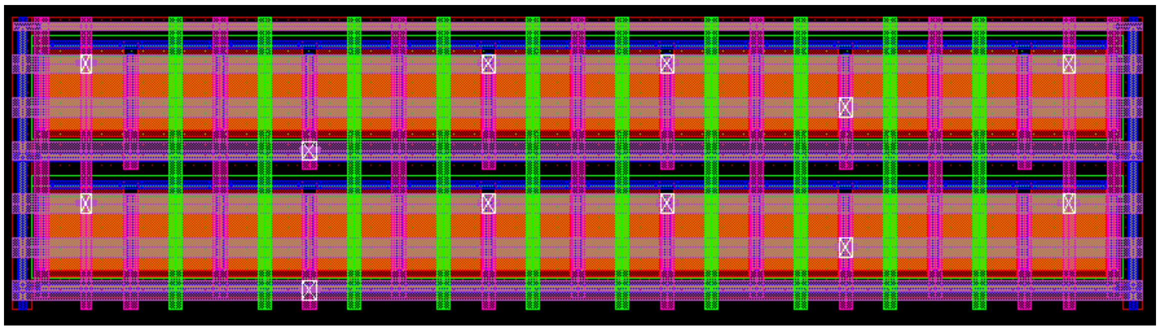
Second, in the phase of drawing the horizontal metal rails (drains local routing) mentioned in 5.7 we do the same thing: giving each metal rail the name of the device that is supposed to be connected to it , so now we have metal extensions and the horizontal metal rails drawn over the gates with the same name ,The remaining task is to connect these two nets to each other's by VIAS .

In order to place the VIAS we must know the exact location of the intersections between these two metals to place the VIA where it belongs.

It's a heavy task and took a lot of time to search for a method to get these locations easily so we can place the vias without any problems.

With the help of Eng. Fady and Eng. Islam (silicon vision layout and CAD team) we managed to find a function that can get these locations very easily (checkvia).

This function is just like magic it searches for every intersection between different metal layers with the same net name and return the bounding box of this intersection which is the exact



location of the VIA that must be placed.

Figure 55: Vias

The next step is saving the locations of all the vias in multidimensional matrix and looping over these locations to put the vias.

Then with the same procedure we can put the vias between the local drain connections in 5.7 and the global drain connections 5.8 (vias from metal 3 to metal 4).

The following figure shows the CM array after being fully connected:

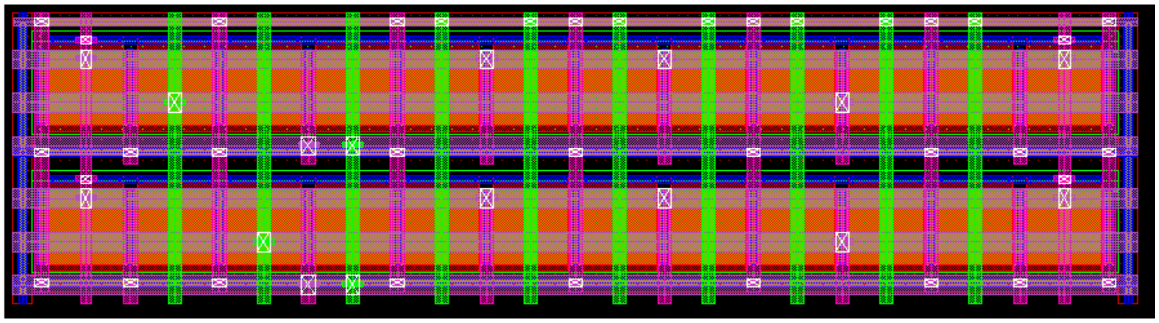
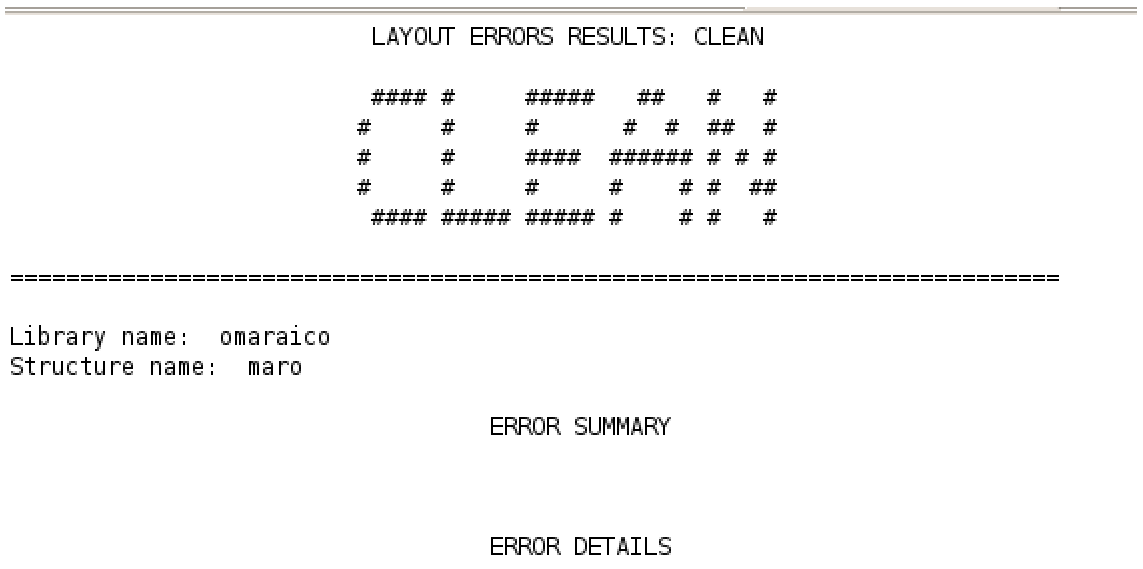


Figure 56: Fully routed CM

5.11. Verification

After the routing process is fully performed, physical design checks (DRC, ERC, LVS) are performed to make sure that our layout follows the design rules, in order to be manufacturable and to make sure that the layout follows the schematic.



```
                         LAYOUT ERRORS RESULTS: CLEAN

                          #### #     #####   ##   #   #
                         #     #     #      #  #  ##  #
                         #     #     ####  ###### # # #
                         #     #     #     #    # #  ##
                          #### ##### ##### #    # #   #

===========================================================================

Library name:  omaraico
Structure name:  maro

                                  ERROR SUMMARY



                                  ERROR DETAILS
```

Figure 57: Physical verification results

These checks were done and reported a clean Layout design.

Chapter 6.
Results

6.1. Tool overview

In this chapter we will show the results of our work as a tool called ACML that do all the flow mentioned in the previous chapters in few steps.

First the user has a schematic and wants to generate the layout for this schematic, the first thing the user must do is to source the code in synopsis console window.

After sourcing the code a window will appear and ask the user to select the transistors of the mirror that the user want to generate its layout as shown in the first figure in the example below.

After that the tool extracts the parameters of this mirror (width per finger , length per finger) and gives the user some options to edit as (tie width ,minimum DRC rules of metals and other layers) , it also give the user options to select whether the layout out will be source shared which means if s\he wants he unit of matching as single finger or two fingers and also s\he want to select the option of width division which gives the tool a permission to divide the width of each unit and double the number of units to get the best matching pattern available , this window as shown in the second figure in the next example.

After finishing this step and selecting “OK”, a window appear to the user asking the user to enter the current of each device that will affect the placement and routing of each device and shows different pairs that can be generated from this schematic, each pair with its area (width and length) and the excess space needed for routing if it exists.

These pairs are arranged from the best case to the worst one (that has a bad aspect ratio or a lot of dummies that consume a lot of area that is considered wasted), the user has the option to selected one pair from them that suits his/her top cell floor-planning, this window as shown in the third figure in the next example.

After finishing the selection of the pair the layout will be generated in a few seconds with clear physical checks (DRC, LVS)

6.2. Tool’s detailed output

1. Test case #1

In this section we will show some examples of the steps mentioned above and the generated layouts.

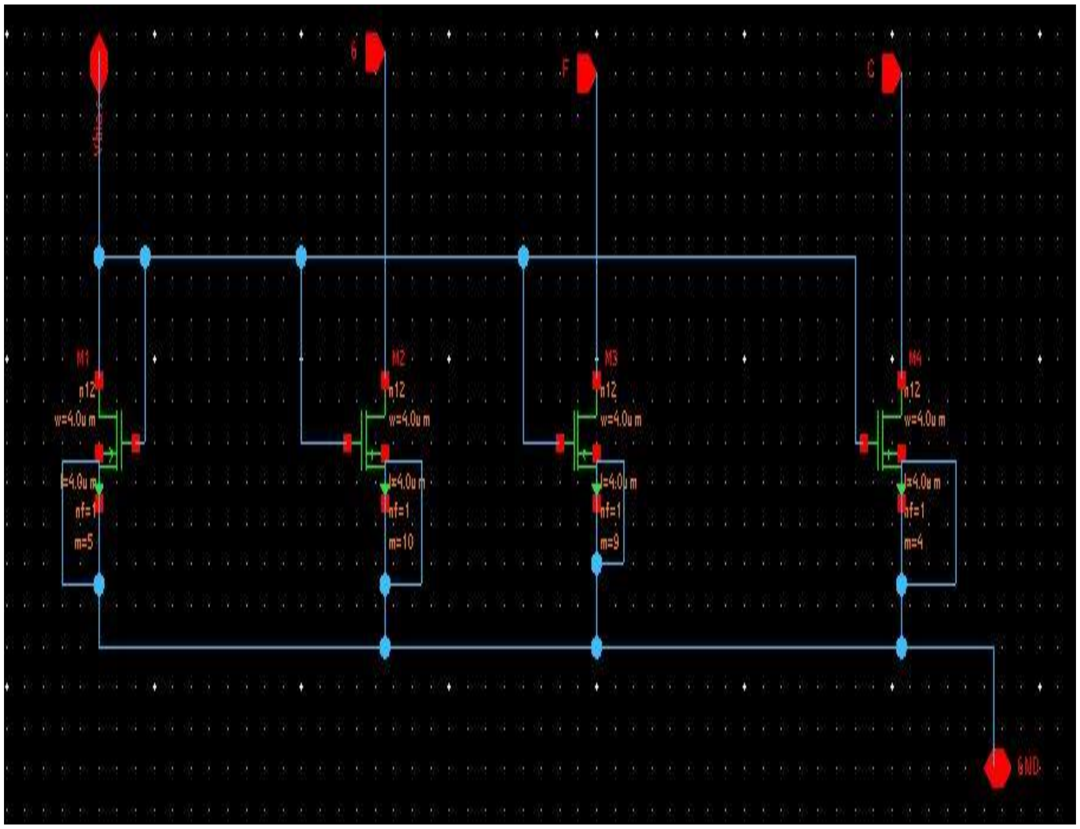


Figure 58 : Given schematic of current mirror

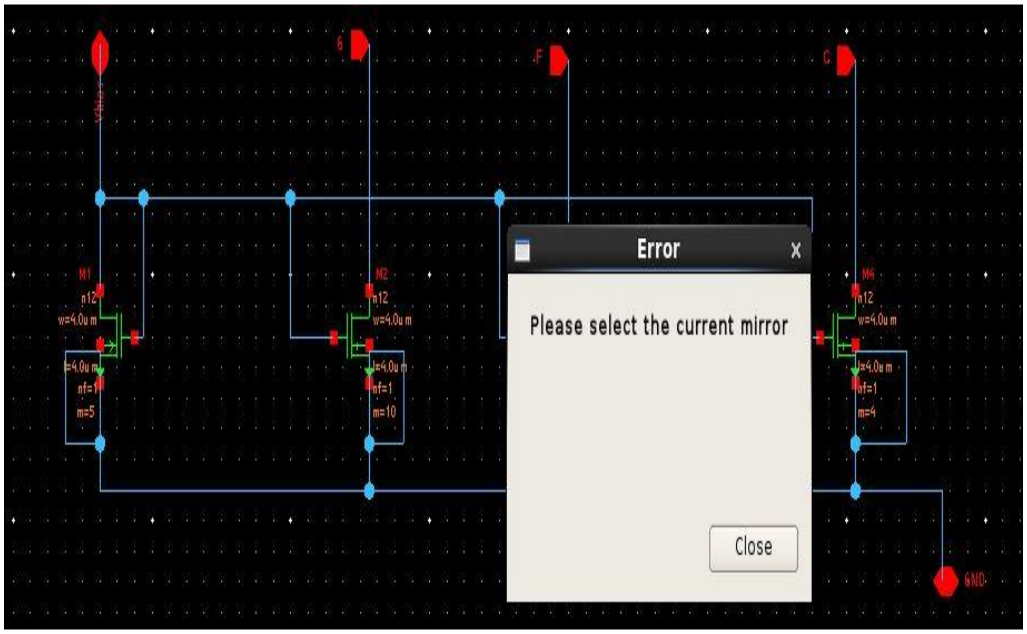
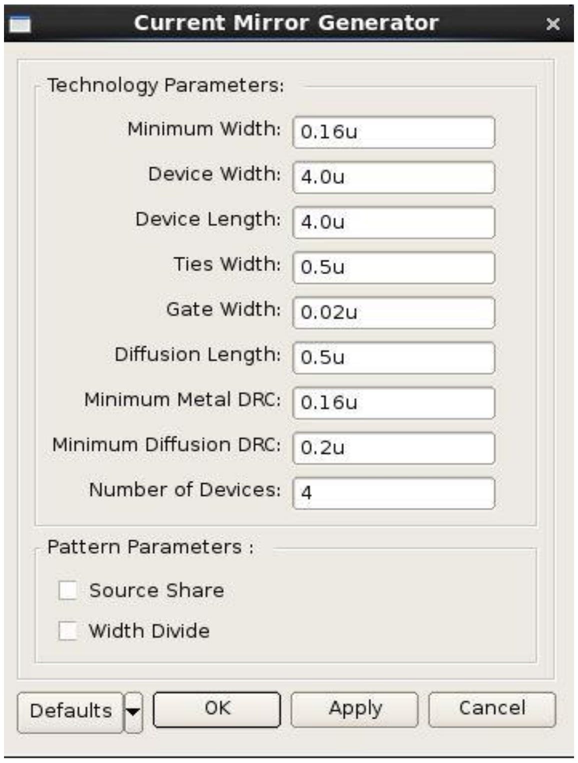


Figure 59 : window appeared asking to select the transistors of the mirror



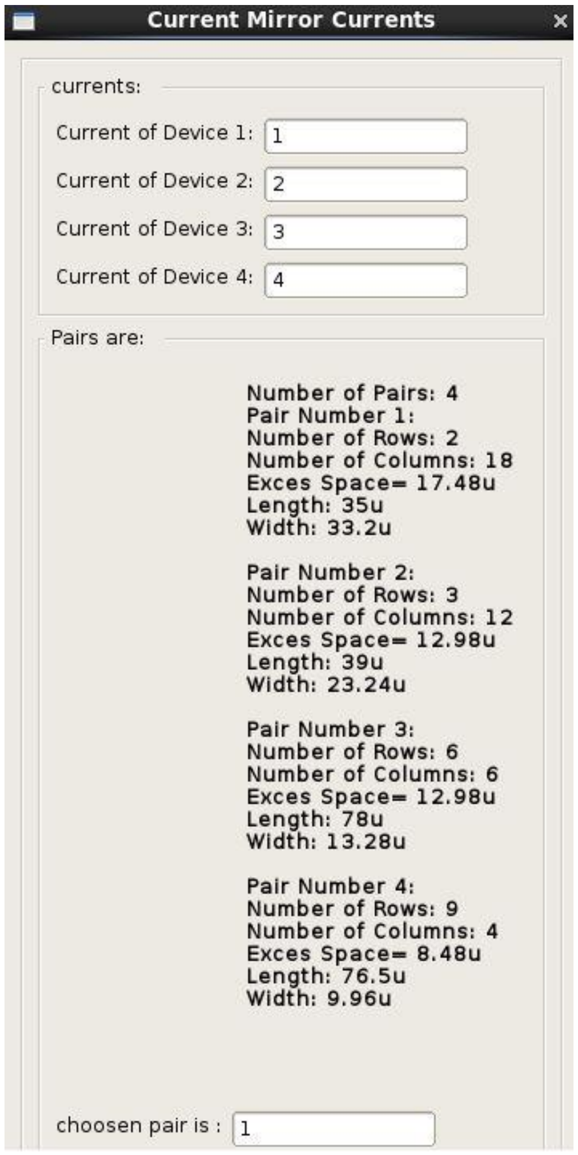
Technology Parameters:

Minimum Width:	0.16u
Device Width:	4.0u
Device Length:	4.0u
Ties Width:	0.5u
Gate Width:	0.02u
Diffusion Length:	0.5u
Minimum Metal DRC:	0.16u
Minimum Diffusion DRC:	0.2u
Number of Devices:	4

Pattern Parameters :

- [ ] Source Share
- [ ] Width Divide

Figure 60 : window after selecting the transistors of the mirror


Current Mirror Currents

currents:

Current of Device 1:	1
Current of Device 2:	2
Current of Device 3:	3
Current of Device 4:	4

Pairs are:

```
Number of Pairs: 4
Pair Number 1:
Number of Rows: 2
Number of Columns: 18
Exces Space= 17.48u
Length: 35u
Width: 33.2u

Pair Number 2:
Number of Rows: 3
Number of Columns: 12
Exces Space= 12.98u
Length: 39u
Width: 23.24u

Pair Number 3:
Number of Rows: 6
Number of Columns: 6
Exces Space= 12.98u
Length: 78u
Width: 13.28u

Pair Number 4:
Number of Rows: 9
Number of Columns: 4
Exces Space= 8.48u
Length: 76.5u
Width: 9.96u
```

choosen pair is : 1

Figure 61 : window shows all the available pairs for the user to select one from them

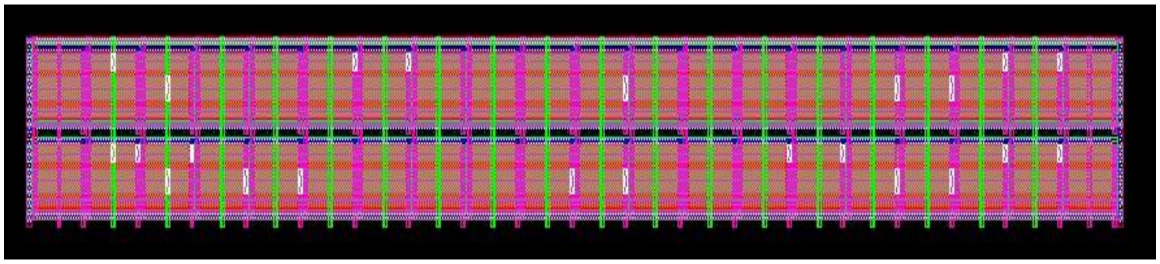
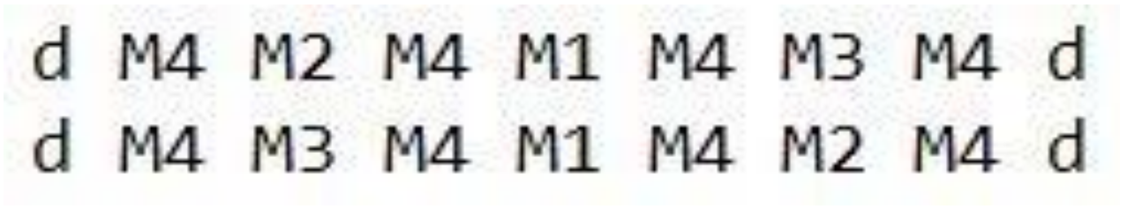


Figure 62 : Generated layout if the user chose pair 1



```
d M4 M2 M4 M1 M4 M3 M4 d
d M4 M3 M4 M1 M4 M2 M4 d
```

Figure 63 : matching pattern for this case

2. Test case #2

Here is another example from the tool

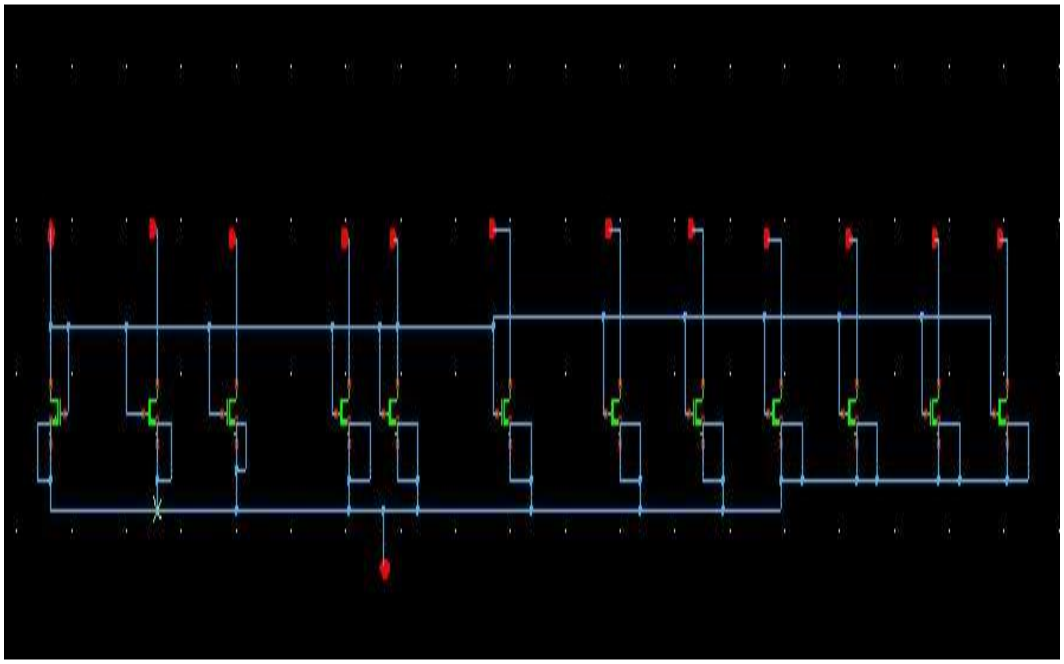
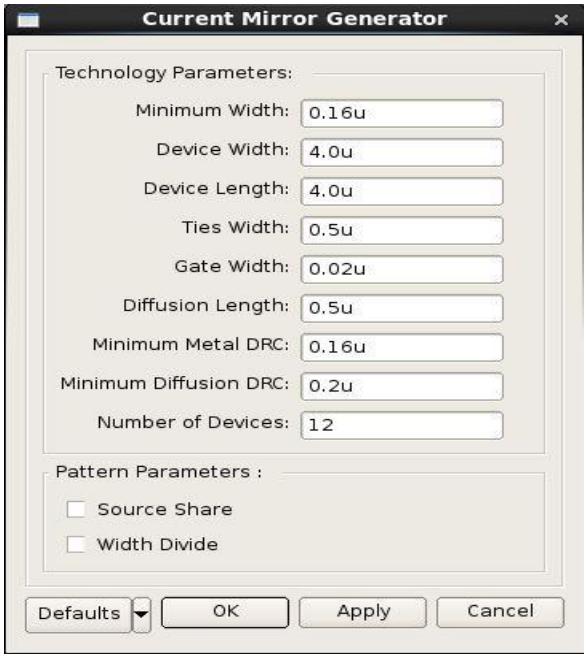


Figure 64 : the given schematic for case 2



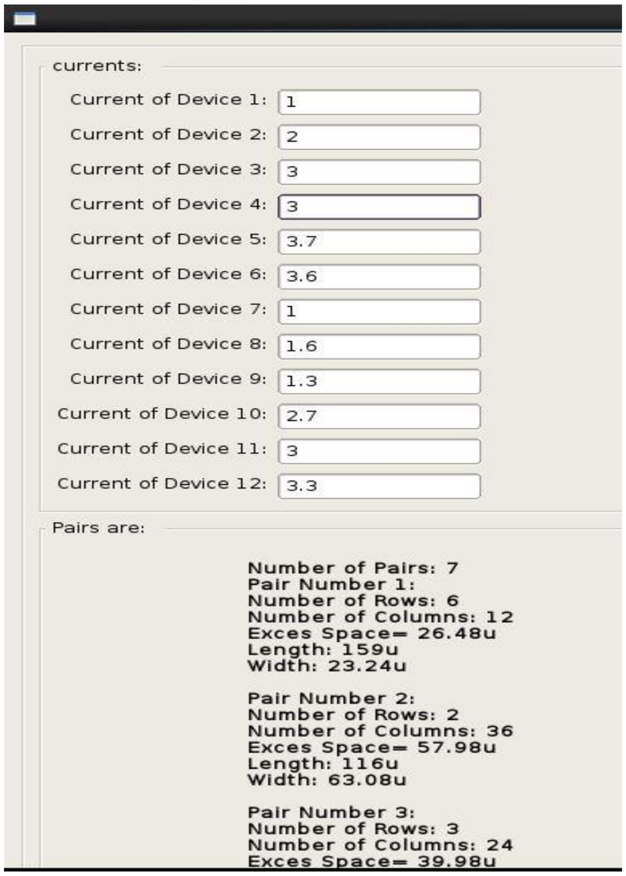
Technology Parameters:

Minimum Width:	0.16u
Device Width:	4.0u
Device Length:	4.0u
Ties Width:	0.5u
Gate Width:	0.02u
Diffusion Length:	0.5u
Minimum Metal DRC:	0.16u
Minimum Diffusion DRC:	0.2u
Number of Devices:	12

Pattern Parameters :

☐ Source Share
☐ Width Divide

Figure 65 : parameters of devices for case 2



currents:

Current of Device 1:	1
Current of Device 2:	2
Current of Device 3:	3
Current of Device 4:	3
Current of Device 5:	3.7
Current of Device 6:	3.6
Current of Device 7:	1
Current of Device 8:	1.6
Current of Device 9:	1.3
Current of Device 10:	2.7
Current of Device 11:	3
Current of Device 12:	3.3

Pairs are:

```
Number of Pairs: 7
Pair Number 1:
Number of Rows: 6
Number of Columns: 12
Exces Space= 26.48u
Length: 159u
Width: 23.24u

Pair Number 2:
Number of Rows: 2
Number of Columns: 36
Exces Space= 57.98u
Length: 116u
Width: 63.08u

Pair Number 3:
Number of Rows: 3
Number of Columns: 24
Exces Space= 39.98u
```

Figure 66 : available pairs with their area for case 2

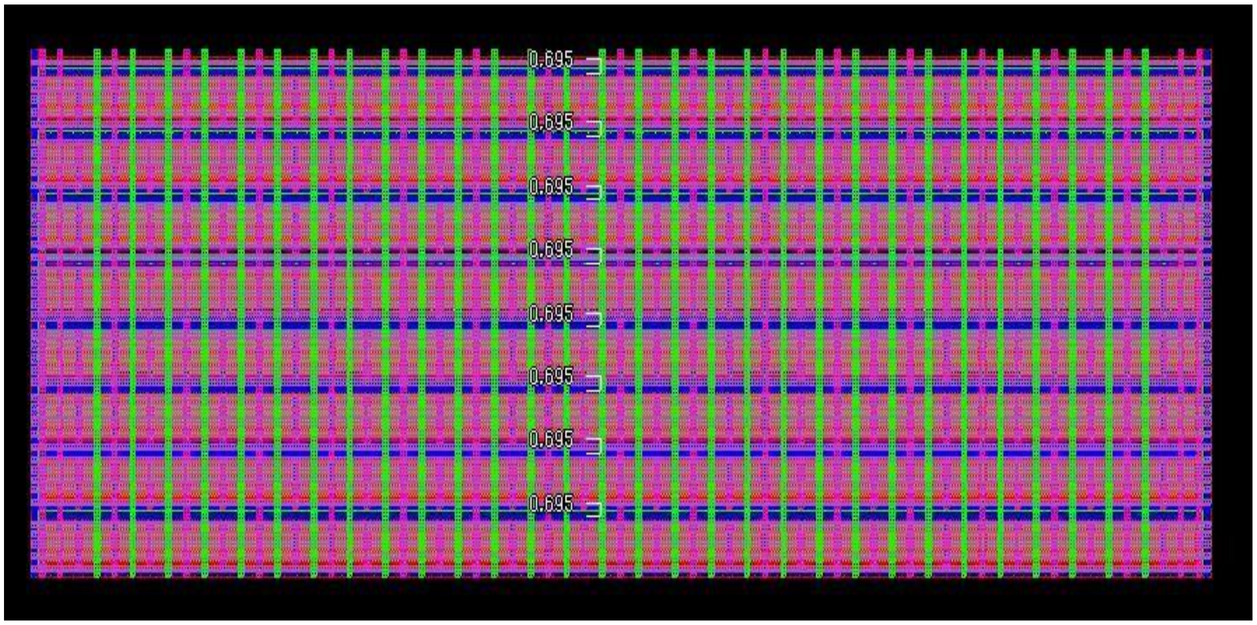
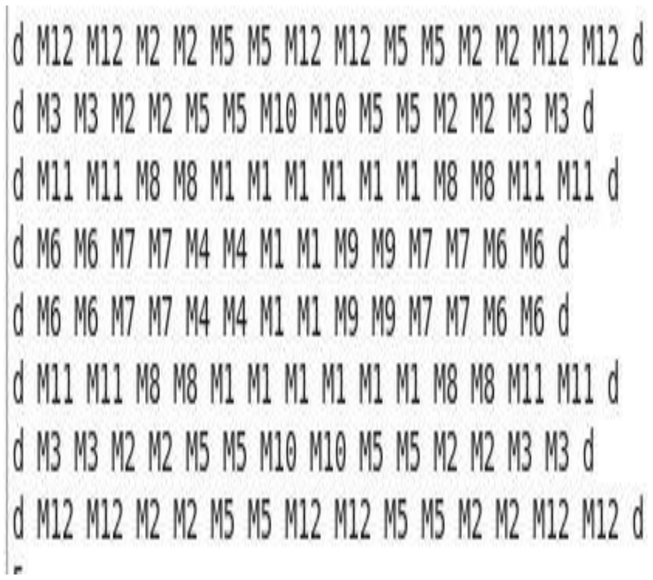


Figure 67 : generated layout for case 2



```
d M12 M12 M2 M2 M5 M5 M12 M12 M5 M5 M2 M2 M12 M12 d
d M3 M3 M2 M2 M5 M5 M10 M10 M5 M5 M2 M2 M3 M3 d
d M11 M11 M8 M8 M1 M1 M1 M1 M1 M1 M8 M8 M11 M11 d
d M6 M6 M7 M7 M4 M4 M1 M1 M9 M9 M7 M7 M6 M6 d
d M6 M6 M7 M7 M4 M4 M1 M1 M9 M9 M7 M7 M6 M6 d
d M11 M11 M8 M8 M1 M1 M1 M1 M1 M1 M8 M8 M11 M11 d
d M3 M3 M2 M2 M5 M5 M10 M10 M5 M5 M2 M2 M3 M3 d
d M12 M12 M2 M2 M5 M5 M12 M12 M5 M5 M2 M2 M12 M12 d
```

Figure 68 : selected pattern for case 2

Chapter 7.
Conclusion and future work

7.1. Conclusion

Analog layout automation is an important step as it has an impact on the analog design. Analog layout automation would decrease the headache from manual layout and decrease the effort of finding the optimum design. The process flow of layout generation and verification as follows. First, we have the schematic and constrains. Then, floor planning, routing, physical verification, PEX, and finally post layout simulation to ensure a working design after the added parasitic from the generated layout. Several layout generators is made throughout history. For example, ILAC, ALSYN, LAYLA, ALG, AIDA-L, exc.

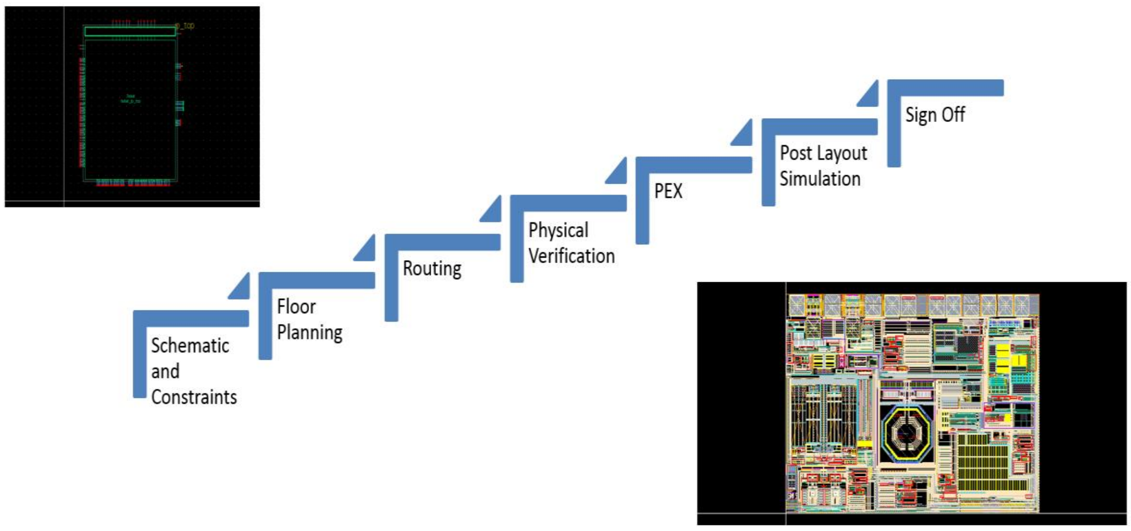


Figure 69: Layout: from specs to sign off

Our Tool ACML generates a powerful and efficient layout tested by the engineers at SI-VISION company a leading company in the field of electronics and after a few modifications they will use our tool in their industry as it produces the needed layout in a very small time compared to do manually and as the circuits size increase the decrease in time increase exponentially as we test this process by testing a current mirror already done at the company by a layout engineer in a full day work (around 8 hours) our tool made it in just one click (10-15 seconds) and this save the time for engineers to focus on the complicated circuits that is not yet can be layed out automatically .

Our tool takes into consideration all the layout problems as LATCH UP & ELECTROMIGRATION and solve each one them during the process.

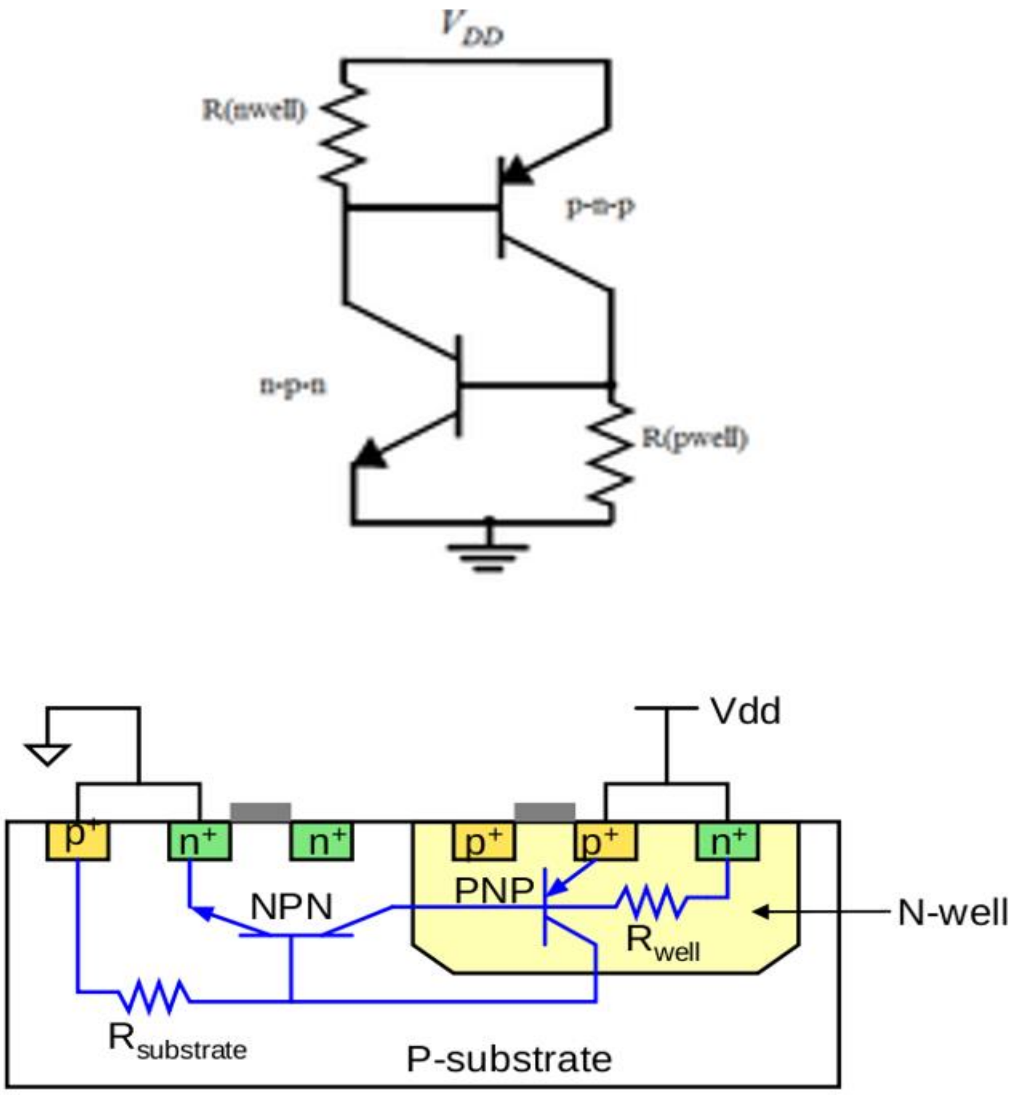


Figure 70: Latchup

We solve latch up by putting the bulk ties between the rows and also around the circuits to decrease the contact resistance (Rsubstrate and Rwell)so the bulk is truly tied to ground in case of nmos and to the supply in case of pmos so the positive feedback will not begin as there is no voltage drop will be created on the base of the BJT transistor and to solve electromigration we made the routes of widths corresponds to the current that will flow on them to avoid voids that can be made inside routes due to high current.

Our technical analyses and experimental results have confirmed its competition over some existing automation tools and expert handcrafted layouts.

7.2. Future work

Our tool can integrated with other blocks to be more powerful, time saving tool.

1) Block recognition

   In out tool we select the transistors that build our current mirror manually to run the tool on them but for a further enhancement we can build a code that detect the Current mirror automatically from the netlist generated from the schematic so we can run the code on the whole schematic that may contain other circuits like differential pairs, vco , regulators and any other block and the code extract the transistors that make the current mirror automatically .

2) Global routing

   Our tool route the current mirror block that we generate its layout only but as an enhancement we can make a code that route the blocks together in code to automate the full process and to save time with

3) Layout multiple current mirrors at the same time

   Out tool now can generate the layout of a single selected current mirror but as an enhancement we can make it generate the layout of multiple circuit at the same time `in order to same time of the engineers

4) Choose the pair according to the floorplan

We now choose the pair we want from the second pop up window but as an enhancement and future work we can create an algorithm that chose the best pair according to the area that fits the available space in the floorplan

References

[1] M. Bojarski et al., "End to End Learning for Self-Driving Cars", 2016.

[2] K. He et al., "Deep Residual Learning for Image Recognition", 2016.

[3] "ImageNet Large Scale Visual Recognition Competition (ILSVRC)", Image-net.org, 2017. [Online]. Available: http://www.image-net.org/challenges/LSVRC/.

[4] K. Simonyan and A. Zisserman, "VERY DEEP ConvOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION", 2014.

[5] H. Li et al., "Acceleration of Deep Learning on FPGA", 2017.

[6] M. Peemen et al., "Memory-Centric Accelerator Design for Convolutional

[7] "Learning about deep learning", Recode, 2017. [Online]. Available: https://www.recode.net/2016/5/4/11634228/learning-about-deep-learning.

[8] "FPGA Acceleration of Convolutional Neural Networks - Nallatech", Nallatech, 2017. [Online]. Available: http://www.nallatech.com/fpga-acceleration-convolutional-neural-networks/.

[9] B. Moons et al., "Energy-Efficient ConvNets Through Approximate Computing", 2016.

[10] D. Patterson et al., Artificial neural networks. Singapore: Prentice Hall, 1996.

[11] Y. Qiao, J. Shen, T. Xiao, Q. Yang, M. Wen and C. Zhang, "FPGA-accelerated deep convolutional neural networks for high throughput and energy efficiency", Concurrency and Computation: Practice and Experience, vol. 29, no. 20, p. e3850, 2016.

[12] C. Zhang et al., "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks", 2015.

[13] "Deep Learning 101 - Part 1: History and Background", Beamandrew.github.io, 2018.[Online].Available:https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html.

[14] "alexnet_tugce_kyunghee.pdf - ImageNet Classification with Deep Convolutional Neural Networks Alex Krizhevsky Ilya Sutskever Geoffrey E Hinton Presented", Coursehero.com, 2018. [Online]. Available: https://www.coursehero.com/file/24481166/alexnet-tugce-kyungheepdf/. [Accessed: 11-Jul- 2018].Neural Networks?", 2017.

[15] T. ARTICLES, N. PRODUCTS, G. ELECTRONICS, C. PROJECTS, E. MICRO, V. Lectures, I. Webinars, I. Training, P. Search, T. DB, B. Tool, R. Designs and S. H.L., "Purpose and Internal Functionality of FPGA Look-Up Tables", Allaboutcircuits.com, 2018. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/purpose-and-internal-functionality-of-fpga-look-up-tables/. [Accessed: 11- Jul- 2018].

[16] Indico.desy.de, 2018. [Online]. Available: https://indico.desy.de/indico/event/7001/session/0/contribution/1/material/slides/0.pdf. [Accessed: 11- Jul- 2018].

[17] E. Nurvitadhi et al., "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?", 2017.

[18] V. Sze et al., "Efficient Processing of Deep Neural Networks: A Tutorial and Survey", 2017.

[19] Y. Chen, T. Krishna, J. Emer, and V. Sze, “Eyeriss : An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks Future of Deep Learning Recognit ion DCNN Accelerator is Crucial • High Throughput for Real-time,” IEEE Int. Solid-State Circuits Conf. , Feb. 2016.

[20] "An Intuitive Explanation of Convolutional Neural Networks", the data science blog, 2018. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. [Accessed: 10- Jun- 2018].

[21] "Understanding Convolutional Neural Networks for NLP", WildML, 2018. [Online]. Available: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. [Accessed: 10- Jun- 2018].

[22] "Activation Functions: Neural Networks – Towards Data Science", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 11- Jun- 2018].

[23] "CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2018. [Online]. Available: http://cs231n.github.io/neural-networks-1/. [Accessed: 11- Jun-2018].

[24] "A Quick Introduction to Neural Networks", the data science blog, 2018. [Online]. Available: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/. [Accessed: 11- Jul- 2018].

[25] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", 2012.

[26] "Why dropouts prevent overfitting in Deep Neural Networks", Medium, 2018. [Online]. Available: https://medium.com/@vivek.yadav/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701. [Accessed: 11- Jul- 2018].

[27] "Caffe | Deep Learning Framework", Caffe.berkeleyvision.org, 2018. [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed: 11- Jul- 2018].

[28] "ImageNet Large Scale Visual Recognition Competition (ILSVRC)", Image-net.org, 2017.[Online].Available:http://www.imagenet.org/challenges/LSVRC/2012/nonpub-downloads.

[29] "pmgysel/alexnet-forwardpath", GitHub, 2018. [Online]. Available: https://github.com/pmgysel/alexnet-forwardpath. [Accessed: 12- Jul- 2018].

[30] "Strategies for pipelining logic", Zipcpu.com, 2018. [Online]. Available: http://zipcpu.com/blog/2017/08/14/strategies-for-pipelining.html. [Accessed: 11- Jul-2018].

[31] E.A Vittoz , Basic Analog Layout Techniques, August 2015 [Online].

[32] Prof. Guoxing Wang, Layout for Analog Integrated Circuits, [Online].

[33] Alan Hastings, Texas Instruments, An introduction to Matching and Layout

[34] Julius Georgiou, MOS Transistor Matching, Dept. of Electrical & Computer Engineering, University of Cyprus, Nicosia