MODULE 4 - PART B

LAYOUT AND LAYOUT TOOLS

In MODULE 4 Part A, we had learnt how to draw the LAYOUT of a CMOS INVERTER and some BASIC DESIGN RULES.

For your PROJECT you will be drawing these types of LAYOUTS for different kinds of STANDARD CELLS like INVERTERS, NAND and NOR GATES and so on.

What is a STANDARD DIGITAL CELL?

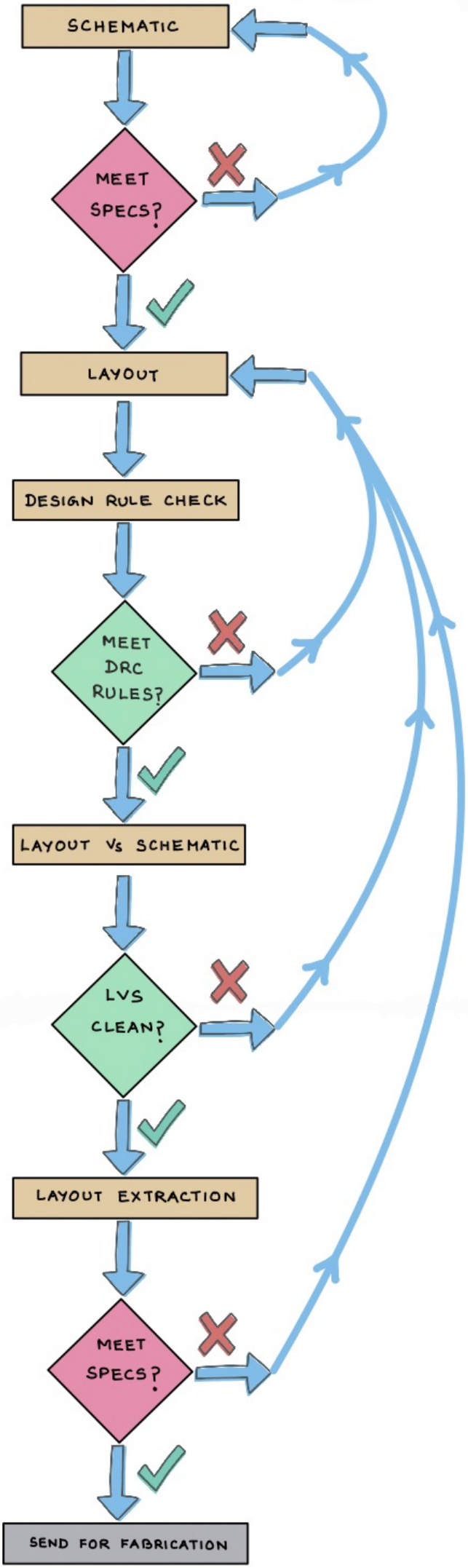
A STANDARD DIGITAL CELL is a library of digital logic gates (for example, INV, NAND2, NAND3, NOR2 etc.) which are used as basic building blocks for designing more complex circuits.

We have seen one standard cell in the last part (Part A) of Module 4, which is an INVERTER.

So in the DESIGN CHAIN, we first draw the SCHEMATIC then we do the, SIMULATIONS to MEET THE SPECIFICATIONS (for Power, Delay etc.). After this we draw the LAYOUT. Then we do the rule checking often called the, DESIGN RULE CHECK (DRC).

DRC ensures that our layout design is manufacturable. The DRC rule file is provided by the foundry.

Complex designs invariably suffer from design and design entry errors. The DRC can help us fix these errors so that the



design can be fabricated / manufactured after the DRC is 'CLEAN', that is all errors have been removed and cleared, we move to the next step,

LAYOUT VS SCHEMATIC (LVS)

LVS will ensure that our layout is the same as our schematic. That is, LVS will make sure,

ⓐ All logic gates are connected to the correct corresponding INPUT / OUTPUT / V_{DD} / GND. This ensures functional correctness i.e. if the logic is right or not.

ⓑ The width and the length of transistors between the schematic and the layout are the same. This ensures the same electrical performance between the schematic and the layout. (We know that the current is proportional to the (Width / Length) and hence, if the widths and the lengths are different, then the currents will be different)

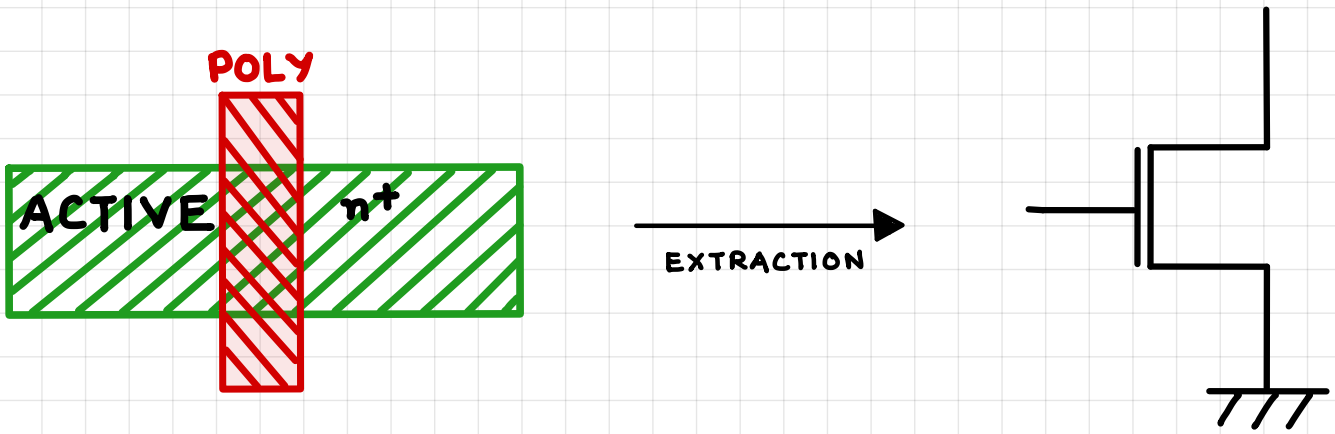
Please note that these checks are more or less an automated process and there are commercially available tools that do this.
The foundry provides the files for the DESIGN RULE CHECKS.
The DRC and LVS steps allow us to detect errors in the schematic and the layout.
After the LVS is clean, we will move to the next step which is,

LAYOUT EXTRACTION

The layout extraction will extract a schematic view of the layout with all the parasitics. The LAYOUT process introduces some additional resistors and capacitors called parasitics and this may change the specifications of our circuit. So we continue to simulate the EXTRACTED schematic and check if the specifications are met and then make appropriate changes to the layout.
This is done iteratively until all the required specifications (power, delay etc.) are met.
Thus, the extracted view allows us to understand the role of the parasitics in our design.
We can say that,
Circuit extraction extracts a schematic representation of a layout including transistors, interconnects and possibly wire and device resistances and capacitances

For example, if we have the layout below, it will be extracted to an nMOS with the appropriate resistances,

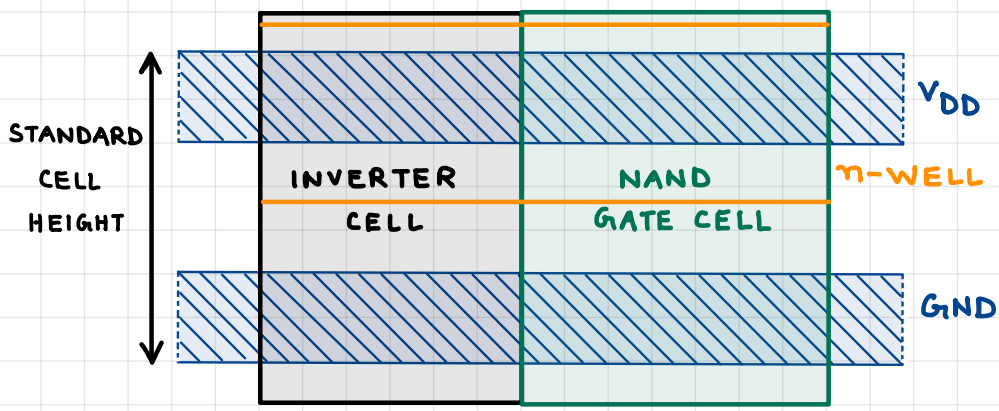
capacitances etc. and we can find the corresponding current - voltage characteristics.



Now, before we move on to more complex layouts, we will briefly discuss the STANDARD CELL DESIGN approach.

- STANDARD DIGITAL CELLS are designed for a standard cell height or pitch.

This means that, when we had drawn our inverter cell with a a certain pitch between the V_{DD} and GND as below, if we now draw another gate (for example, NAND or NOR), we can abut it with our inverter cell and use the same continuous V_{DD} and GND tracks. Thus, digital cells are designed for a standard cell height or pitch.



- Cells with multiple W_p / W_n are designed to meet power performance specifications.

  We will see this later in Modules 5 and 6.

- A large design is hierarchically decomposed into smaller designs and eventually to standard cells.

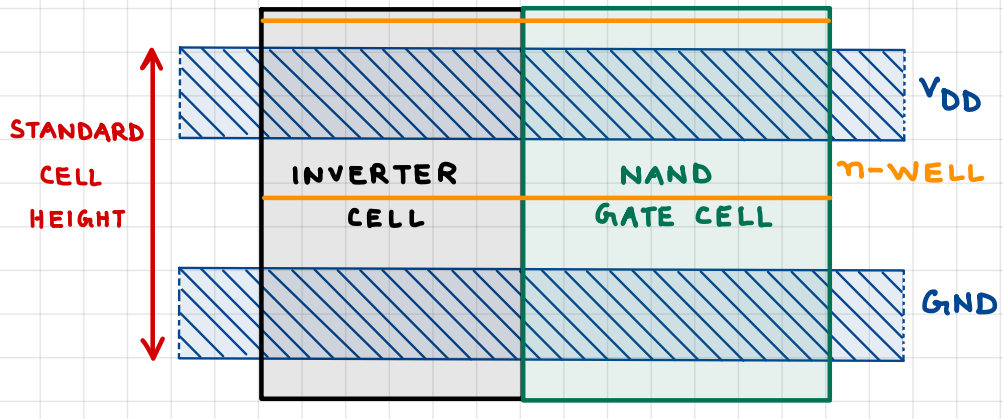
  For example, if we want to design an ALU, we will break it up into the Arithmetic Unit (AU), Logic Unit (LU) and the Shift Unit (SU) and again further break these units into gates or standard cells (NAND, NOR, INV etc)

  This decomposing of a large design is mostly an automated process today and several tools exist that will help us acheive this.

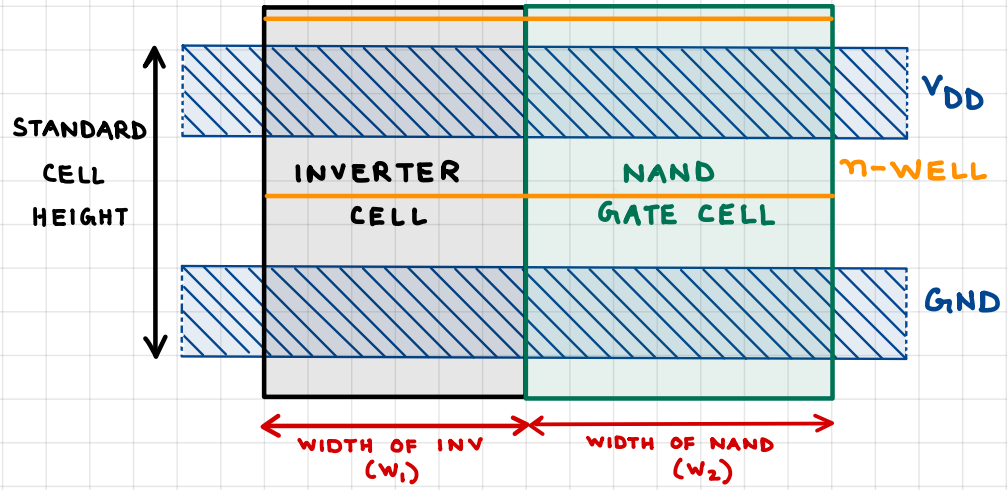
We now know that, a STANDARD CELL LAYOUT ⟶ is the layout of a digital cell (INV, NAND2, NAND3, NOR2 etc.) which follows some COMMON RULES.

We will now see what these rules are.

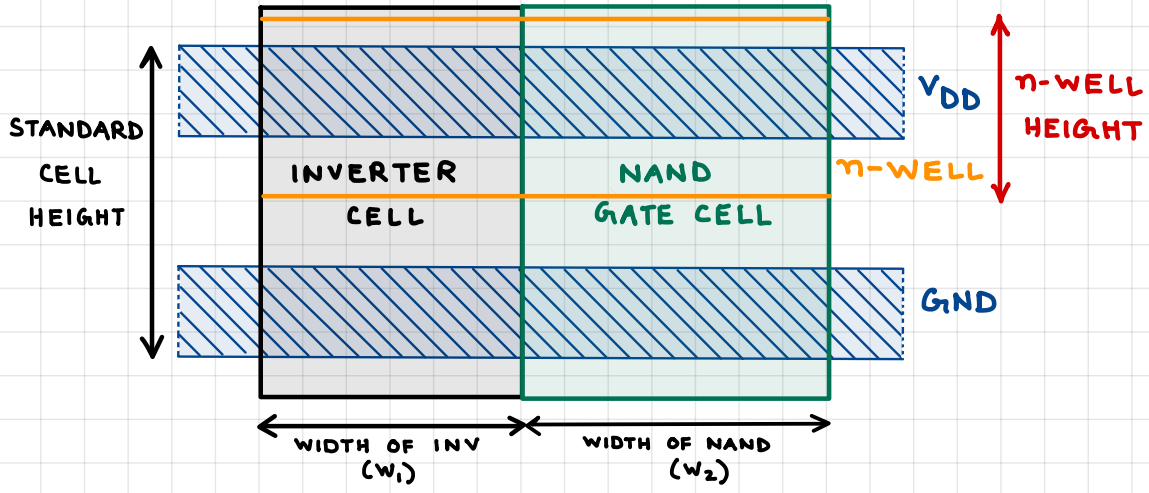
① In the library of standard cells, all the cells have the same HEIGHT (which is the distance between the V_{DD} / V_{CC} and GND/V_{SS})



② The WIDTH of the standard cell can vary from one cell to another.



③ The n-WELL is the region where all the pMOS transistors are built. In the standard cell, we typically have the n-WELL on the top. The n-WELL HEIGHT is IDENTICAL for all standard cells.

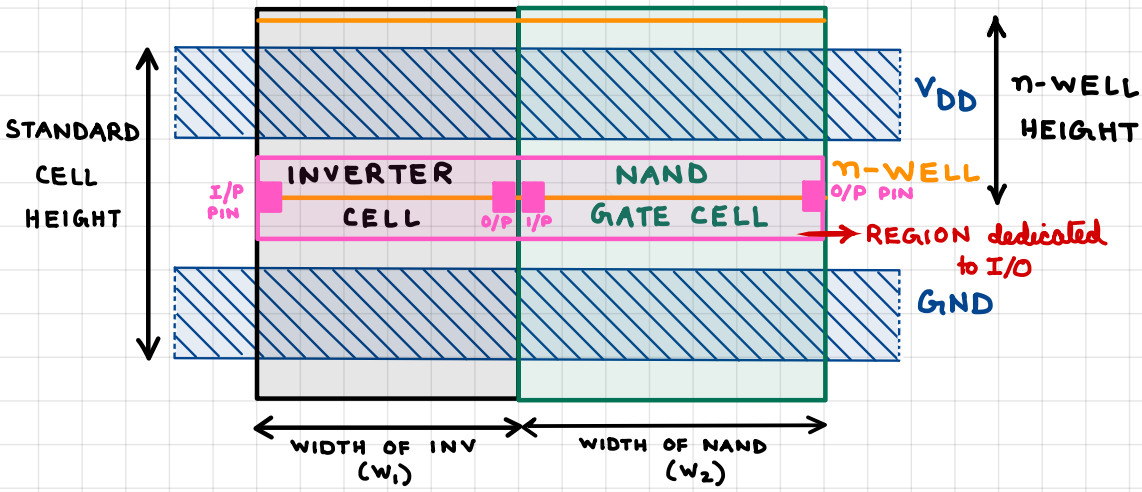


The boundary of the standard cell can vary along the x-axis as the width of the standard cell can vary.

Note that the STANDARD CELL RULES do not tell us where to draw the transistors or the poly within the transistors.

④ Another important feature of the STANDARD CELL is the location of the inputs and the outputs. Typically, in a STANDARD CELL, the inputs and the outputs go out from the center. This is the REGION dedicated to the I/O (in pink below)

For example, the input pin can be on the left side and the output pin can be on the right side and so on.



⑤ Standard cells typically use local metal lines (M1 and M2 only) and the higher layer metals are not used.

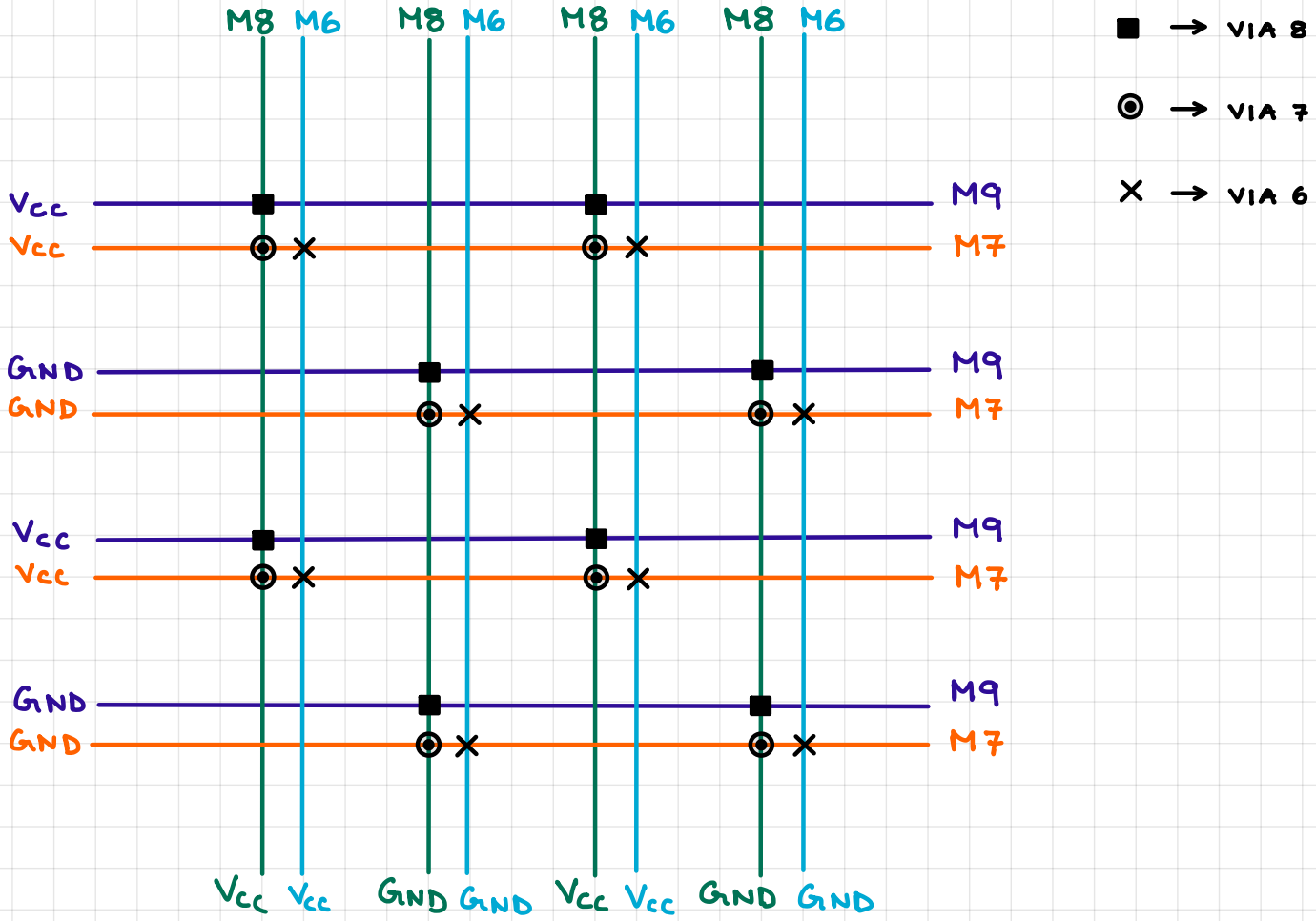
- Local Metal lines are M1, M2, M3 and M4 in modern processes.
- Intermediate Metal lines are M5 and M6 and finally,
- Global Metal lines are M7, M8 and M9.

Since, VLSI designs are hierarchical, we use,

- Local Metal lines M1 and M2 ⟶ to route standard cells
- Local Metal lines M3 and M4 ⟶ to connect multiple standard cells in a functional unit. For example, when we build an adder or a multiplier.

- Intermediate Metal lines M5 and M6 ⟶ to connect multiple functional units. For example, when we want to build an ADDER + MULTIPLIER or a SMALL ALU.
- Global Metal lines M7, M8 and M9 ⟶ for global connections across the chip. For example, when we want to connect memory and logic and external I/O and so on.

The other important thing to remember is that, POWER and GROUND, that is V_{DD} and GND are routed at all metal levels to reduce resistance on the power lines. Power and GND are routed from M9 (or, whatever the highest layer is) all the way down to M1.
This is done in a very special way. If the highest metal line (say, M9) is drawn in the horizontal direction, carrying V_{DD} and GND alternately, then the next highest metal line (M8) will run in the orthogonal direction (vertically). These metal layers will of course be connected with VIAS (via 8 in this case). So the connections to the chip will be made from the top metal layer (M9) and then it will go down to the next lower layer (M8), then the power will be routed to the next lower layer (M7) which will again be in the horizontal direction, connected through VIAS (via 7) with M8, and so on.
(see figure below)



Thus, Power lines alternate between V_{DD} and GND.

Alternate layers are drawn in orthogonal directions (this is mostly for convenience, so that routing is easier)

Signal lines are the metal lines that are a part of the interconnections. These can be drawn in either direction on any metal level. However, modern processes require that signal lines are also orthogonal in different layers. This means that if M1 is horizontal, M2 runs vertical, M3 is horizontal and so on. This keeps our design modular, clean and easy to debug.

This is how the STANDARD CELL is built and the STANDARD CELL LIBRARY is developed. We put together several standard cells to create functional units like adders and multipliers

and again these functional units are put together to create larger units like ALUs and eventually connect this to memory to create a complete DATAPATH.

We have seen that as circuit designers, we first draw the SCHEMATIC and then we draw the complete LAYOUT (with clean DRC and LVS)
However, in between these two steps, there is another intermediate step which is very handy. This is the STICK DIAGRAM.
In this step, we draw the LAYOUT in the form of sticks.
This is very useful in helping us figure out how the structure will look like.
We will now look at how a STICK DIAGRAM is drawn and how we can optimize a STICK DIAGRAM.

STICK DIAGRAM :-

1. A stick diagram is a graphical view of a layout
2. It is simply a way of floor-planning a circuit in preparation for a layout. (So that all the cells can be reached and connected easily and power can be provided)
3. Sticks are a means of capturing the topography (which is the connection between gates or standard cells) and the layer information (i.e. where the poly, diffusion etc. are located)
4. Sticks convey layer information through color codes.
(You will need to use multiple color pens to be able to

draw stick diagrams. Sometimes grayscale is used but that is much harder to draw)

⑤ Stick diagrams act as an interface between symbolic logic (Boolean expressions or schematic) which is a mathematical description AND the actual layout which is a physical description.

Thus, a STICK DIAGRAM is said to be a cartoon of the LAYOUT.

Stick diagrams do not show many of the details, such as :-

① Exact transistor placements
② Transistor sizes
③ Transistor widths, boundaries, Design Rules etc.

What stick diagrams do show is the topographic information i.e. which elements are connected together, how the different layers are placed and so on.

As we have discussed before, we use different colors to denote different layers in stick diagrams.
Commonly used colors for the different layers are shown below :-

COLOR INDEX :-

- BLUE - METAL 1
- BLACK - METAL 2
- RED - POLYSILICON
- YELLOW - P-DIFFUSION
- GREEN - N-DIFFUSION

(Sometimes green is used for both p and n diffusion. However, we are going to use different colors)

- RED DASHED - WELL (for stick diagrams)

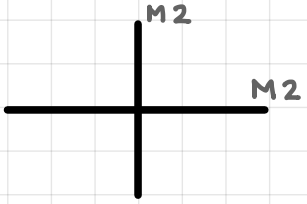
X, ● or ■ - CROSS or BOX or ROUND - CONTACTS or VIAS

Let us now look at the RULES that we must follow when drawing STICK DIAGRAMS.

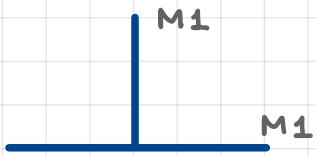
RULES :-

RULE 1 ⟶ When two or more sticks of the same type cross or touch, they represent an electrical contact.

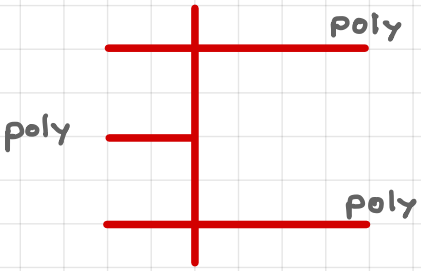
Examples :-



Here, M2 crosses M2 and the cross over point is considered a SHORT.



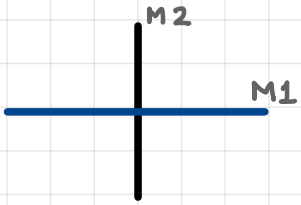
Here, the point where M1 touches M1, that point is considered a SHORT.

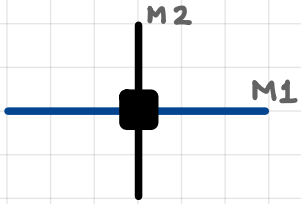


Here, the points where the poly sticks cross or touch each other, those points are considered as SHORTS.

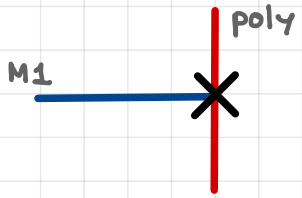
RULE 2 ⟶ When two or more sticks of different types cross or touch, there is no electrical contact unless there is a VIA/CONTACT

Examples :-





Here, in the figure on top, M2 crosses M1 and the cross over point is NOT considered a SHORT, unless there is a VIA as shown in the figure below it.

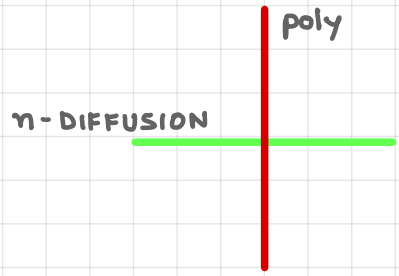


Here, in the figure on top, M1 touches poly and this point is NOT considered a SHORT, unless there is a VIA as shown in the figure below it.

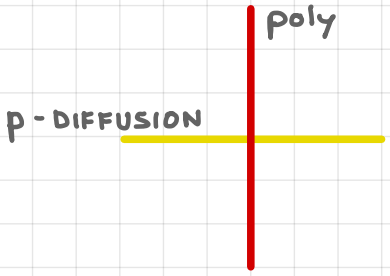
Please note that ●, ■ and ✕ all denote a VIA/CONTACT

RULE 3 ⟶ When a polysilicon crosses a diffusion, it represents a TRANSISTOR.

Examples :-



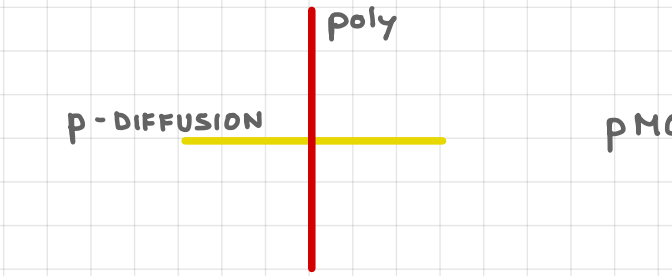
nMOS



pMOS

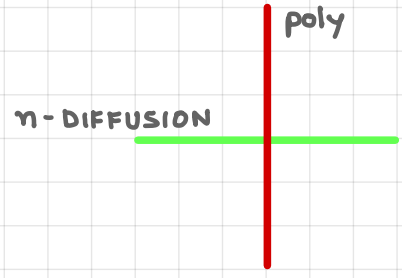
RULE 4 ⟶ In CMOS a demarcation line is drawn to avoid touching of p-diffusion and n-diffusion. All p-MOS must be on one side of this line and all n-MOS must be on the other side of this line. This line represents the n-WELL.

We have seen that the n-WELL is drawn as a red dashed line in stick diagram representations, so we will draw this demarcation line as a red dashed line.



pMOS

line of demarcation representing the n-WELL boundary



nMOS

Now that we know these 4 RULES, let us try to draw the

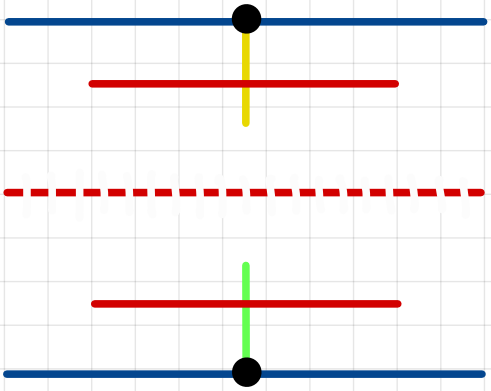
STICK DIAGRAM OF AN INVERTER :-

STEP 1 :- We have V_{DD} on the top and GND at the bottom and these are drawn in metal 1 (M1)

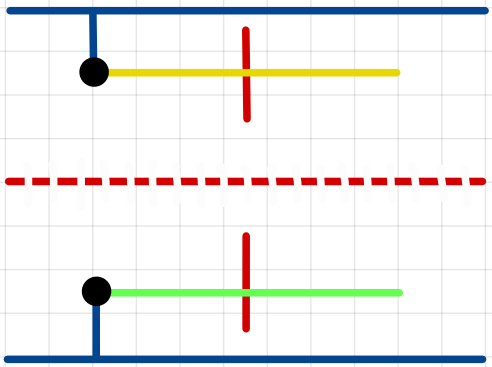
NOTE :- We do not put any labels on the STICK DIAGRAM.

STEP 2 :- We will draw the n-WELL next. As we have seen before, this will be a dashed red line in between V_{DD} and GND.

STEP 3 :- In this step we will draw the transistors i.e. the p-diffusion and poly crossing it (pMOS) and the n-diffusion and poly crossing it (nMOS). Do not forget to draw the contacts of the diffusion to V_{DD} and GND.



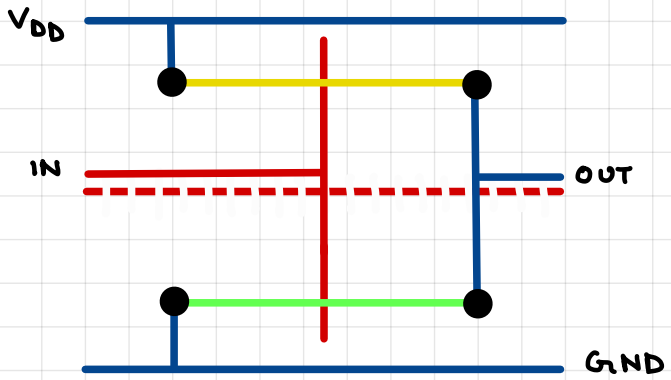
In most modern VLSI layouts the diffusions run horizontally. So, even if the diagram above is electrically correct, we will redraw this to follow the convention as below :-



STEP 4 :- In the last step, we will draw the inputs, outputs and the internal connections.

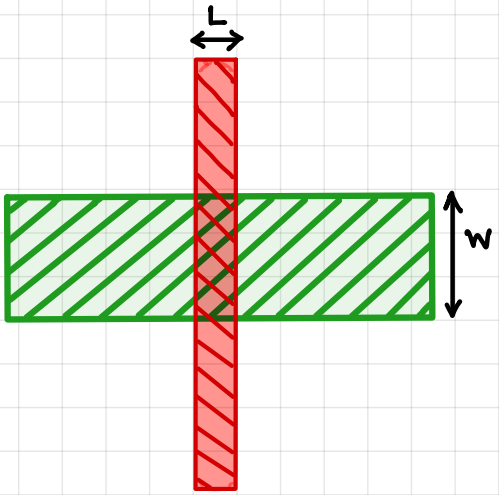
So we will connect the pMOS and the nMOS gates together and this will also be our input to the inverter. This will be made with polysilicon.

The Drains of the pMOS and the nMOS will also be connected together and this will be the output of the inverter. This connection will be made with metal 1 (M1)



Since this is our entire layout for now, we will just label the input, output, V_{DD} and GND. If we were building this as part of the Standard Cell Library, we didn't have to label the V_{DD} and GND.

Let us now look at the layout of a transistor. The length of the polysilicon is the Length (L) of the transistor which is fixed since it is a technology parameter. The width of the diffusion is the width (W) of the transistor which we can change as circuit designers.

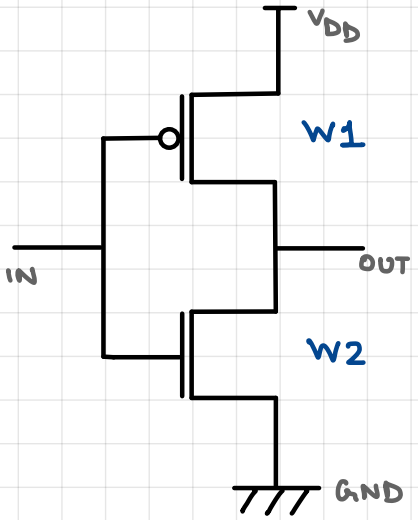


Thus, if W increases, then the width of the diffusion or the green rectangle must increase.

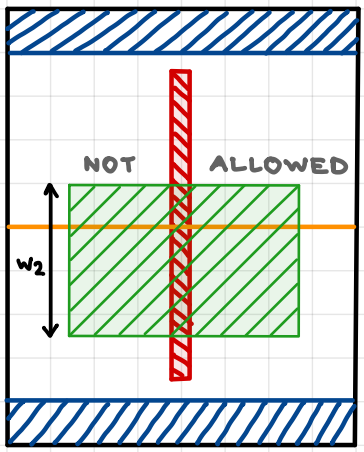
But the question is, can we increase the width of

the diffusion arbitrarily?

Say, we have to draw the layout of the inverter below where the width of the pMOS is W1 and the width of the nMOS is W2 and W2 is very large.

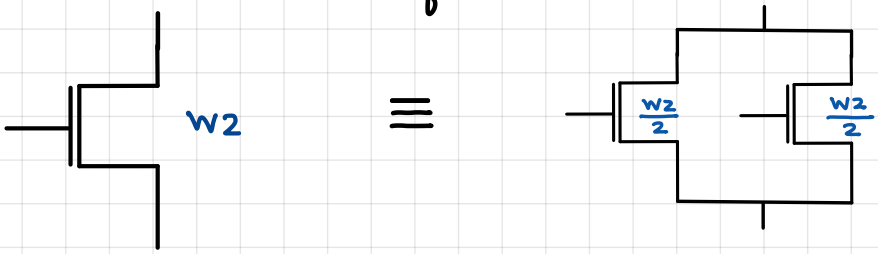


To draw the layout of the nMOS where the width of the diffusion is very large, we can by no means extend the n-diffusion past the n-WELL region because in that case it will no longer remain an nMOS. Thus, the n-diffusion MUST remain within the p-substrate (see layout below)



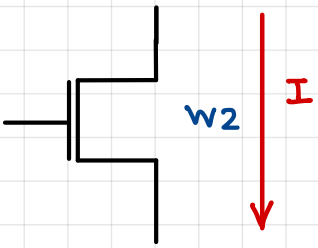
To accomodate this large diffusion width W2, we could potentially increase the height of our standard cell. But that wouldn't be optimal because most of the transistors will have narrower width and that would waste a lot of area. Hence, we must find another way to accomodate a large W2 without increasing the cell height.

To do this, we can split the transistor of width W2 into 2 parallel transistors of width W2/2 each.



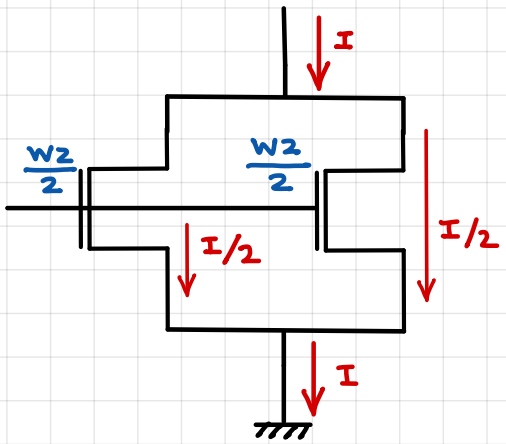
Why are these 2 transistors in parallel and not in series ?

This is because, if the current flowing through the transistor of width W2 is given by I, then



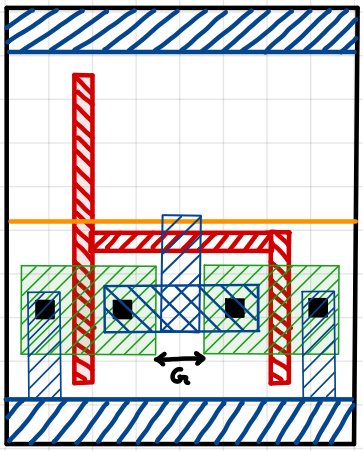
$I \propto W_2$ (from the current equations)

Now, if the transistor width is halved, the current through it will also be half. Therefore, the current flowing through the $W_2/2$ transistors will be $I/2$. Since, we have 2 such transistors in parallel, the total current will be $\frac{I}{2} + \frac{I}{2} = I$.



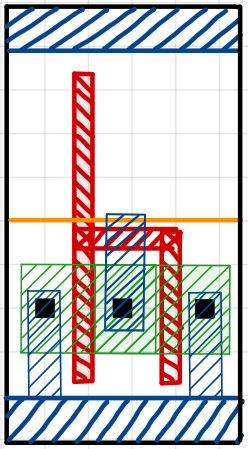
We will now draw the layout of these 2 transistors in parallel

This is shown below.



But now there is this extra gap 'G' we must have between the 2 transistors because of the Design Rule requirements and the extra contact. Thus, the area of our design will be larger.

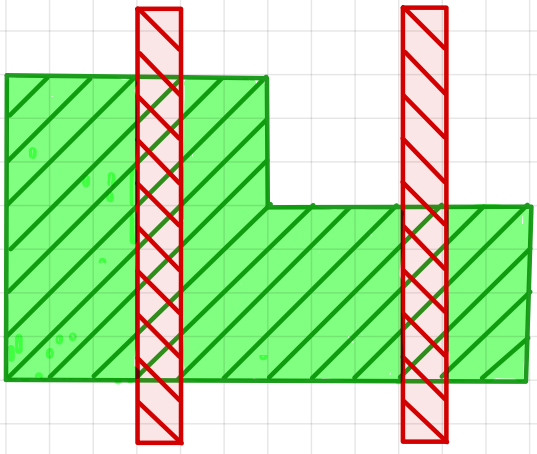
We can solve this problem by realizing that the Drains of these 2 transistors are actually connected too. So, we can



have a continuous diffusion region (known as SHARED DIFFUSION) instead of the two distinct ones that we had before (see figure). This will eliminate the gap 'G' required for the Design Rules and also the extra contact. Because, we can now have a single contact in the continuous diffusion region. As a result, the overall width of our design will be a lot less and hence the area will be smaller.

This is how we can draw the layout of a very wide transistor by splitting it into narrower parallel transistors. This technique is commonly used and each unit of these narrower transistors is known as a FINGER (of the wider transistor). Each of these fingers are of equal width (in this case, $W_2/2$).

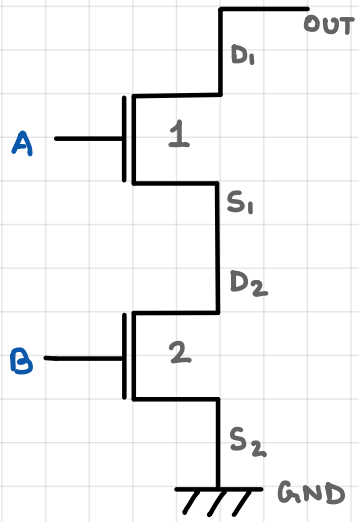
This is because if the widths were unequal our diffusion region would look like this (see figure below) and then the Design Rule Requirements would not be met.



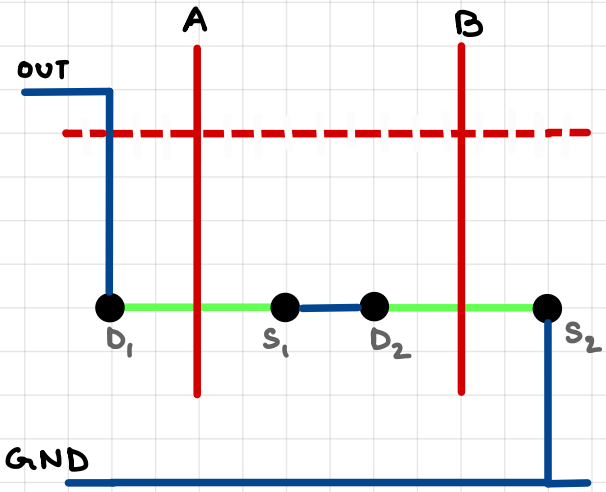
Now, let us look at how STICK DIAGRAMS can be optimized using this concept of shared diffusion.

SHARED DIFFUSION :-

Let us consider the Pull Down Network of a NAND gate.



We will try to understand the concept of shared diffusion using this transistor structure. Please note that we are not drawing the PUN here. We can draw the stick diagram of this structure as below :-

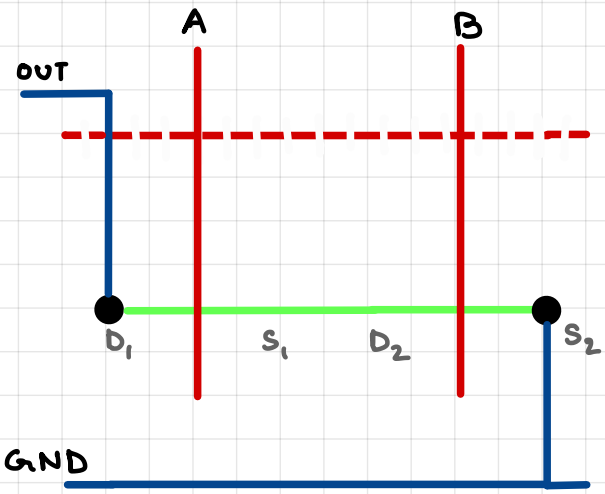


Here we draw the 2 transistors with 2 distinct diffusions and the input polys A and B. The Drain of the transistor 1 is the output of the NAND gate. The Source of this transistor 1 is connected to the Drain of transistor 2 with metal 1 and finally the Source of transistor 2 is connected to GND.

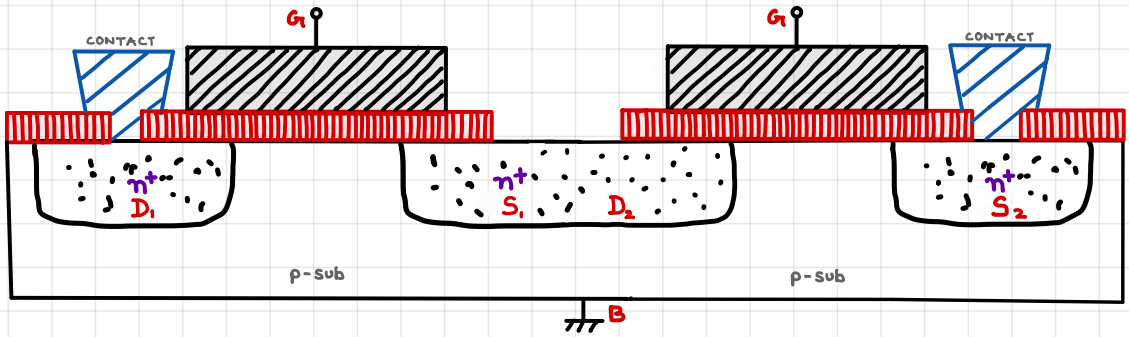
We can see that we again have a gap between the Source of transistor 1 and the Drain of transistor 2. We also have 2 contacts here.

To optimize our cell area, we can use the concept of shared diffusion. Since the S_1 and D_2 are connected, we can have a continuous diffusion here or in other words, share the diffusion. As a result, we can eliminate the 2

contacts between S_1 and D_2 and also eliminate the gap or spacing between them, thus optimizing our cell area. This will also reduce the parasitics in our layout. The optimized STICK DIAGRAM will be drawn as below :-



The side view of this design will look like :-



This optimization process can be formally applied to any complex CMOS gate using the principle of EULER PATHS.

EULER PATHS :-

The principle of EULER PATHS is based on EULER'S work on graph theory. Using this principle we can figure out how to draw circuits with minimum diffusion region.

EULER PATHS are an optimal path through a graph. There are lots of applications of Euler Paths in various fields of engineering. To see how we can use the concept of Euler Paths in Stick Diagrams, let us first define an EULER PATH.

An EULER PATH along a CONNECTED GRAPH is a path that :-

① connects all the vertices

② traverses every edge of the graph only once

CONNECTED GRAPH ⟶ This is a graph where every node is reachable from any other node (NOTE : nodes and vertices are the same thing)

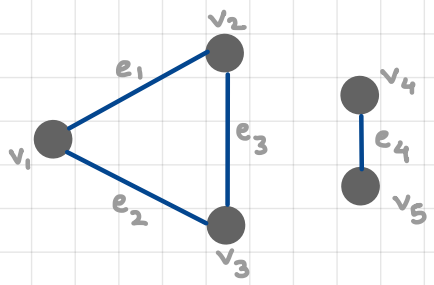
Formally, an EULER PATH can also be defined as :-

A continuous path $P = \{e_1, e_2, \ldots, e_n\}$ in the graph $G(V, E)$, (where V is a set of all vertices $V = \{v_1, v_2, \ldots, v_m\}$ and E is a set of all edges $E = \{e_1, e_2, \ldots, e_n\}$ of the graph) containing every $e \in E$ exactly once.

NOTE : There may be 0 or more than one EULER PATH

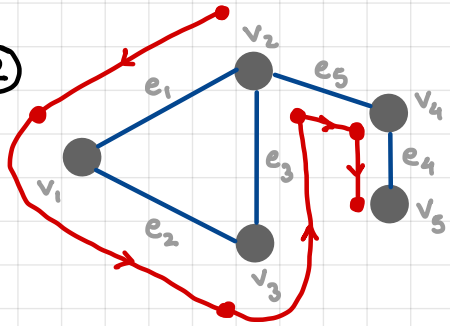
EXAMPLES :-

①



This is NOT a CONNECTED GRAPH.

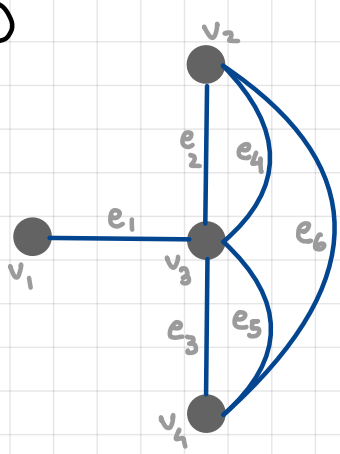
∴ EULER PATH does not exist

② 

Here, we can see that several EULER PATH exists.

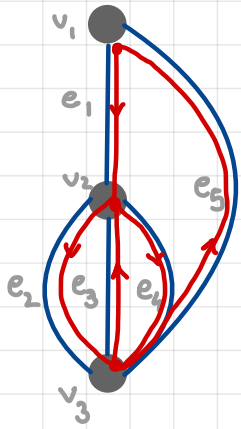
$P_1 = \{e_1, e_2, e_3, e_5, e_4\}$ or

$P_2 = \{e_4, e_5, e_1, e_2, e_3\}$ etc.

③ 

For all paths in this graph, we have to traverse an edge more than once.

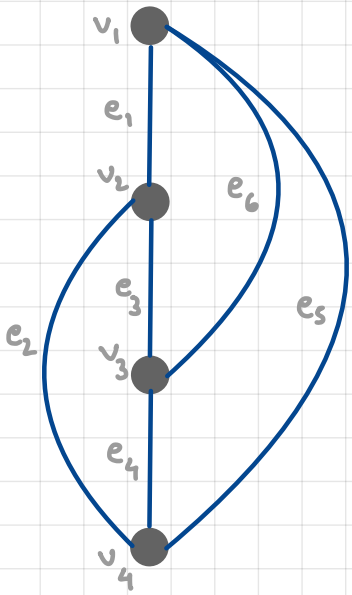
∴ EULER PATH does not exist

④ 

Here too, several EULER PATHS exist.

$P_1 = \{e_1, e_2, e_3, e_4, e_5\}$

$P_2 = \{e_5, e_2, e_4, e_3, e_1\}$ etc.

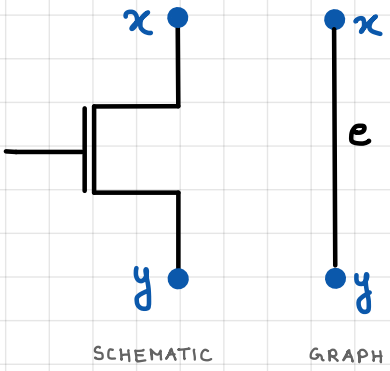
⑤ 

For all paths in this graph, we have to traverse an edge more than once.

∴ EULER PATH does not exist

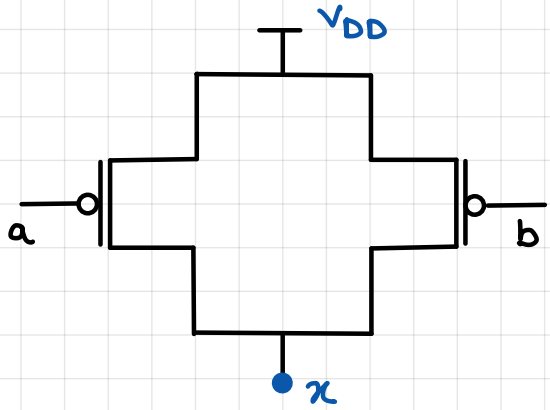
NOTE :- In general, if there are more than 2 vertices with an odd degree (degree of a vertex is the number of edges connected to it), then an EULER PATH does not exist.

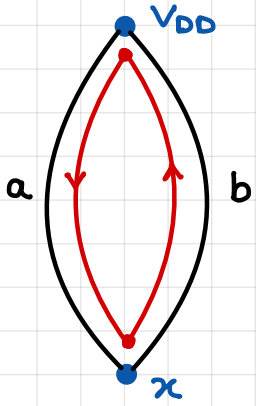
EULER PATH in STICK DIAGRAMS :-



If we have a transistor in a schematic as shown in this figure, between nodes x and y, this is represented as a graph where x and y are the nodes or vertices and the transistor is an EDGE (e)

EXAMPLE :- If we have the following PMOS configuration, then we can represent that as a graph as shown below





Here, the PMOSes a and b are represented as edges in the graph and the nodes V_{DD} and x are the vertices V_{DD} and x of the graph. Here, we have 2 EULER PATHS, $P_1 = \{a, b\}$ and $P_2 = \{b, a\}$

Now, let us look at the steps for drawing the STICK DIAGRAM from a BOOLEAN FUNCTION or SCHEMATIC with minimum diffusion.

STEPS FOR DRAWING THE STICK DIAGRAM :-

(I) Construct a LOGIC GRAPH of the SCHEMATIC for the PUN (Pull Up Network) and the PDN (Pull Down Network) separately.

(II) Construct an EULER PATH for both the PUN and the PDN. Both of these paths MUST BE THE SAME, i.e. the EULER PATH we choose must be common to both the PUN and the PDN.

(III) Sketch the STICK DIAGRAM after having found the common EULER PATH. We will follow the STEPS below :-

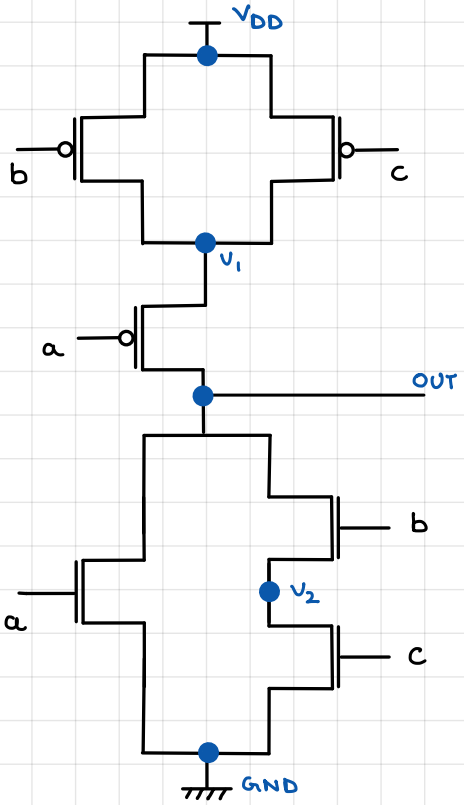
- (a) Sketch the V_{DD} and GND
- (b) Sketch the n-WELL
- (c) The EULER PATH will allow us to make the diffusion continuous. So, we will sketch continuous diffusion lines for n-diffusion and p-diffusion
- (d) Sketch the poly lines corresponding to the inputs of the transistors (in the order of the edges of the EULER PATH) to cross the diffusion lines and create the transistors. Do not forget to make the necessary connections to V_{DD} and GND.

(e) Make the necessary input, output and internal connections to complete the STICK DIAGRAM.

NOTE :- The order of edges in the EULER PATH indicates the order of the transistor gates in the STICK DIAGRAM. This is also the optimal order which eliminates any break in the diffusion (if an EULER PATH exists)

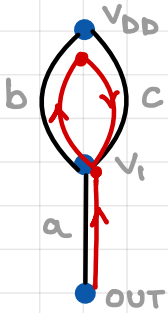
EXAMPLES:

Eg 1 :- Draw the stick diagram corresponding to the following schematic (with minimum diffusion)



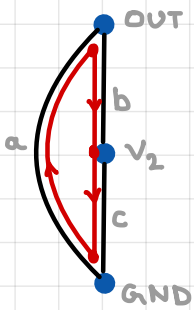
We will first mark all the nodes as shown and find the common Euler Path between the PUN and the PDN.

PUN :-



P(PUN) = {a, b, c}

PDN :-

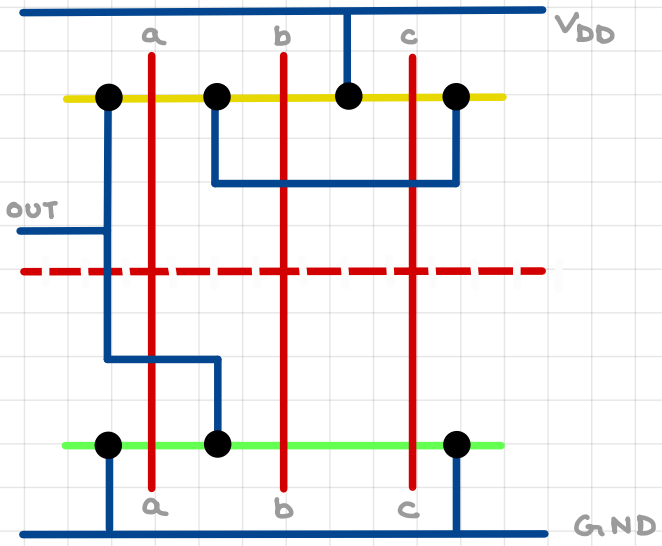


P(PDN) = {a, b, c}

Here, P = {a, b, c} is a common EULER PATH.

(Note that there are other common paths too. e.g. P = {a, c, b})

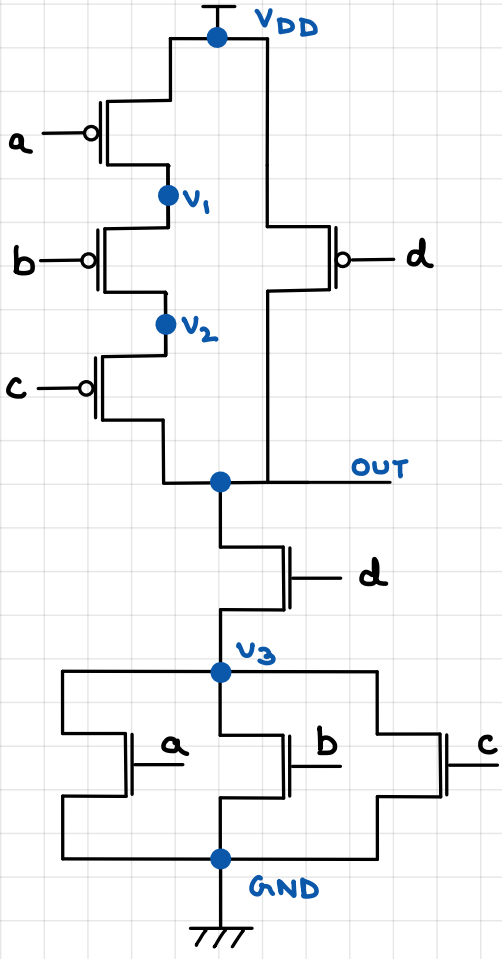
We can draw the STICK DIAGRAM as below :-



Eg 2 :- Draw the stick diagram corresponding to the following Boolean expression (with minimum diffusion)

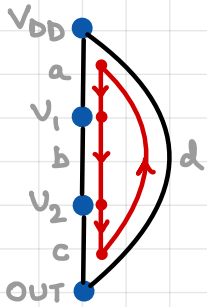
$$OUT = (\bar{a}.\bar{b}.\bar{c}) + \bar{d}$$

We will first draw the schematic corresponding to this Boolean Expression (Please refer to ECE 2020 Class Notes - Module 4 - Part B, if you have forgotten how to do this)



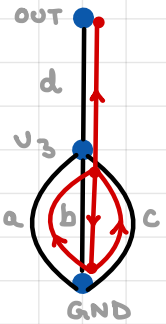
We will first mark all the nodes as shown and find the common Euler Path between the PUN and the PDN.

PUN :-



P(PUN) = {a, b, c, d}

PDN :-

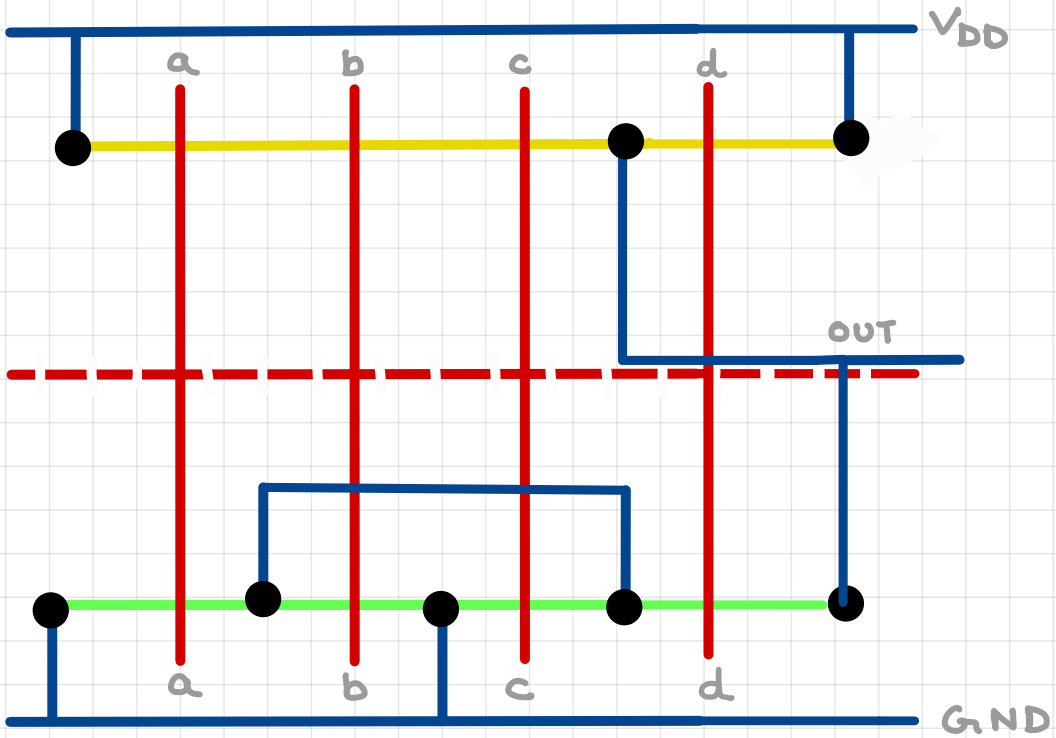


P(PDN) = {a, b, c, d}

Here, $P = \{a, b, c, d\}$ is a common EULER PATH.

(Note that there are other common paths too. e.g. $P = \{b, c, d, a\}$)

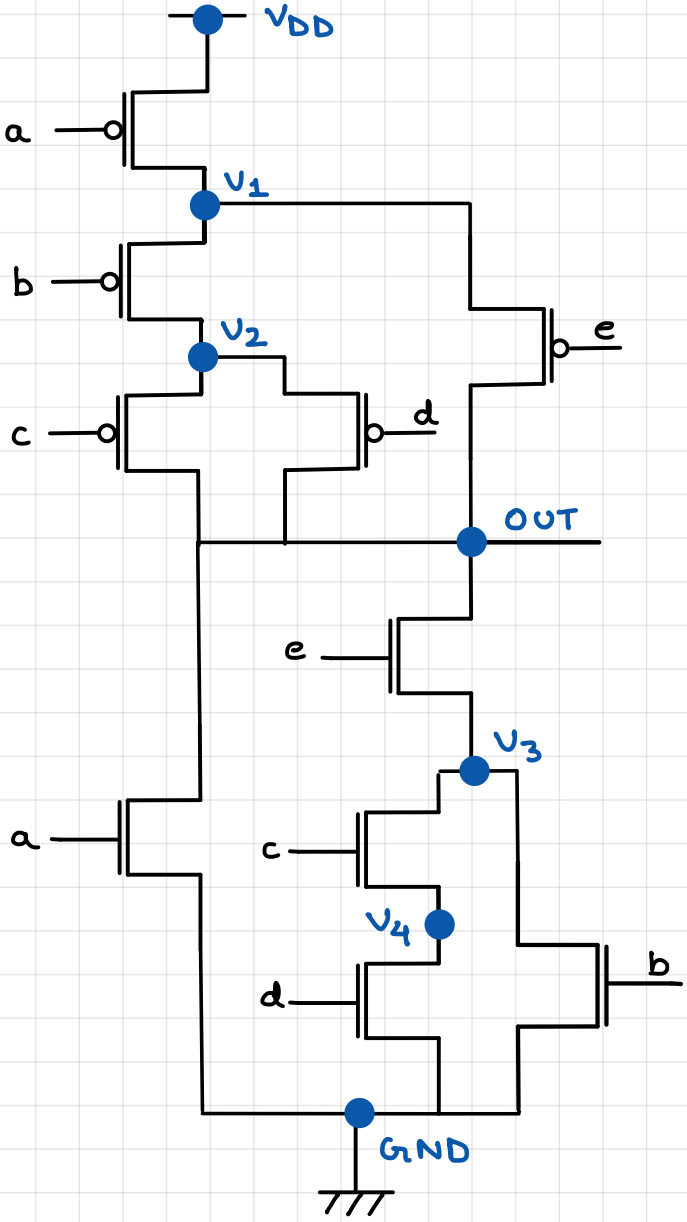
We can draw the STICK DIAGRAM as below :-



Sometimes an Euler Path may not exist (as we have seen before) In such cases a break in the diffusion is needed and cannot be avoided. The idea is to minimize the number of breaks in the diffusion.

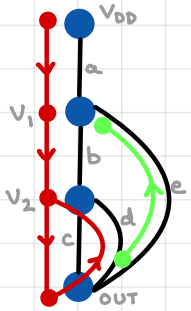
Let us look at one such example.

Eg 3 :- Draw the stick diagram corresponding to the following schematic (with minimum diffusion)



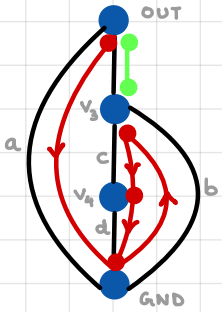
We will first mark all the nodes as shown and find the common Euler Path between the PUN and the PDN.

PUN :-



$\mathcal{P}(PUN) = \{a, b, c, d\} \cup \{e\}$

PDN :-



$\mathcal{P}(PDN) = \{a, b, c, d\} \cup \{e\}$

Here, we do not have any EULER PATHS in the PUN so there will be a break in the diffusion. The transistors a, b, c and d will have shared diffusion and transistor e will be separate. Note that we could have chosen the break in the diffusion to be at some other point too.

We can draw the STICK DIAGRAM as below :-